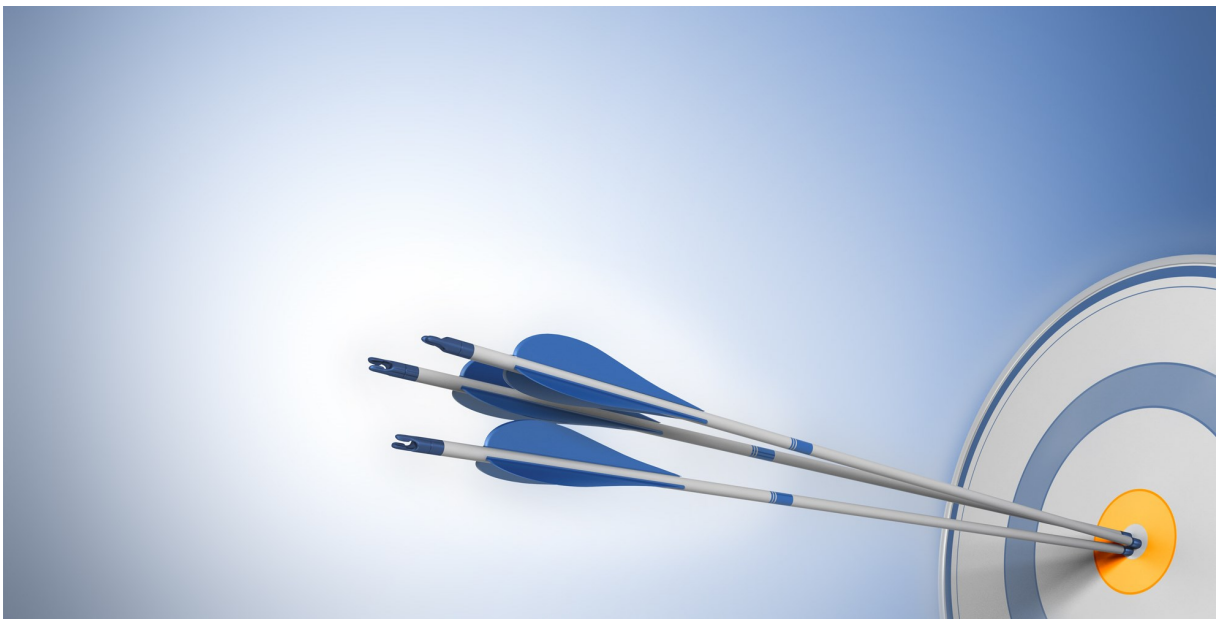


ConSol Software GmbH

ConSol CM Process Designer Handbuch

Version 6.10.5.4

Inhalt

A - Einleitung

In diesem Kapitel werden folgende Themen behandelt:

A.1 ConSol CM für Business Process Management

ConSol CM ist eine **kundenzentrierte Business Process Management Software**. Mit ConSol CM können Sie Geschäftsprozesse kontrollieren und steuern. Im Fokus liegt dabei die menschliche Kommunikation und Interaktion, wie zum Beispiel bei Prozessen im Bereich Helpdesk, Customer Service, Marketing, Vertrieb oder Einkauf. Grundsätzlich lässt sich jeder in einem Unternehmen eingesetzte Prozess mit ConSol CM abbilden und zum Leben erwecken.

Mit ConSol CM können Sie alle in Geschäftsprozessen relevanten Komponenten verwalten und die Prozesse Ihres Unternehmens optimal steuern. ConSol CM wird in verschiedenen Branchen eingesetzt, von Versicherungen und Banken über Modeunternehmen bis zu Herstellern von Fahrkartenautomaten oder Autowaschanlagen. Der flexible Mechanismus zur Prozessgestaltung und die Workflow-Engine bieten die perfekte Grundlage für die Modellierung und Steuerung von unterschiedlichen Geschäftsprozessen.

A.2 Liste der Handbücher

ConSol CM enthält Dokumentation für mehrere Benutzergruppen. Folgende Dokumente sind verfügbar:

- **Administratorhandbuch**
  Ein detailliertes Handbuch für CM-Administratoren über die ConSol CM-Konfiguration mit dem Admin Tool.
- **Process Designer-Handbuch**
  Ein Leitfaden für Workflow-Entwickler über die grafische Benutzeroberfläche des Process Designers und die Programmierung von Workflow-Skripten.
- **Betriebshandbuch**
  Eine Beschreibung der ConSol CM-Infrastruktur, der Serverintegration in IT-Umgebungen und des Betriebs des CM-Systems für IT-Administratoren und -Betreiber.
- **Setup-Handbuch**
  Eine technische Beschreibung des ConSol CM-Setups in verschiedenen IT-Umgebungen. Für Experten der CM-Administration.
- **Benutzerhandbuch**
  Ein Einführung in den ConSol CM Web Client für Endbenutzer.
- **Systemanforderungen**
  Eine Liste aller Voraussetzungen, die für die Installation von ConSol CM erfüllt sein müssen, für IT-Administratoren und CM-Administratoren. Mit jeder ConSol CM-Version veröffentlicht.
- **Technische Release Notes**
  Technische Informationen über die neuen ConSol CM-Funktionen. Für CM-Administratoren und Key-User. Mit jeder ConSol CM-Version veröffentlicht.

A.3 Dieses Handbuch

A.3.1 Bevor Sie dieses Handbuch lesen ...

Wenn Sie dieses Handbuch lesen, setzt Ihr Unternehmen wahrscheinlich ConSol CM als Business Process Management Tool ein und Ihre Aufgabe ist es, das System zu administrieren und die Prozesse Ihres Unternehmens im System zu implementieren. Dieses Handbuch hilft Ihnen dabei, das Prinzip von ConSol CM-Workflows zu verstehen und die Arbeit mit dem Process Designer zu erlernen. Zahlreiche *Tipps und Tricks* von unseren erfahrenen Consultants helfen Ihnen dabei, die beste Vorgehensweise zu finden, wie Sie Ihre Prozesse verbessern können.

Bevor Sie anfangen, mit dem Process Designer zu arbeiten, sollten Sie fundierte Kenntnisse bezüglich der ConSol CM-Administration besitzen, da das Programmieren von CM-Workflows die Nutzung verschiedener CM-Komponenten voraussetzt, die vor oder während der Workflow-Entwicklung konfiguriert werden. Lesen Sie daher zuerst das *ConSol CM-Administratorhandbuch*.

A.3.2 Struktur des Handbuches

1. Zuerst werden einige grundlegenden Komponenten von Geschäftsprozessen allgemein erklärt (siehe dazu diesen Abschnitt).
2. Danach folgt ein Überblick über die Implementierung dieser Prozesse in ConSol CM (siehe Abschnitt Grundkomponenten von ConSol CM-Prozessen).
3. Darauf folgt eine detaillierte Erklärung des Process Designers (siehe Abschnitte Arbeiten mit der Process-Designer-Applikation und Komponenten von ConSol CM-Workflows)
4. Die Abschnitte Prozesslogik, Einführung in die Workflow-Programmierung und Best Practices liefern Expertenwissen über die Entwicklung von Workflows.
5. Da jeder Workflow installiert werden muss, um aktiviert zu werden, wird dieses Thema im Abschnitt Installieren von Workflows behandelt.
6. In den Appendizes finden Sie Listen mit allen wichtigen Begriffen, die in diesem Handbuch verwendet werden (Glossar), mit allen Annotationen (Annotationen) (wichtig für das GUI-Design) und den System-Properties (System-Properties) (wichtig für das CM-Systemmanagement). Beachten Sie auch die Seite mit den Handelsmarken (Marken).

A.3.3 In diesem Handbuch behandelte ConSol CM-Konfiguration

ConSol CM bietet verschiedene Möglichkeiten, kunden- und systemspezifische Funktionen zu implementieren, sodass Sie ein BPM-System erhalten, das genau auf Ihre Kundenwünsche zugeschnitten ist. Ein erheblicher Teil der "Intelligenz" Ihres CM-Systems basiert auf Skripten. Allerdings befinden sich nicht alle Skripte im Process Designer oder sind mit diesem verbunden. Im Process Designer werden nur Skripte gespeichert, die Teil eines Workflows sind. Alle anderen Skripte befinden sich im Admin Tool. Dort können auch bestimmte Workflow-Skripte gespeichert werden.

Daher finden Sie in diesem Handbuch Erklärungen über

- die grafische Gestaltung von Workflows mit dem Process Designer
- das Erstellen von Workflow-Skripten (die entsprechenden Themen sind in der folgenden Grafik hellblau dargestellt)

Eine ausführliche Beschreibung aller anderen Skripte finden Sie im *ConSol CM Administratorhandbuch*.

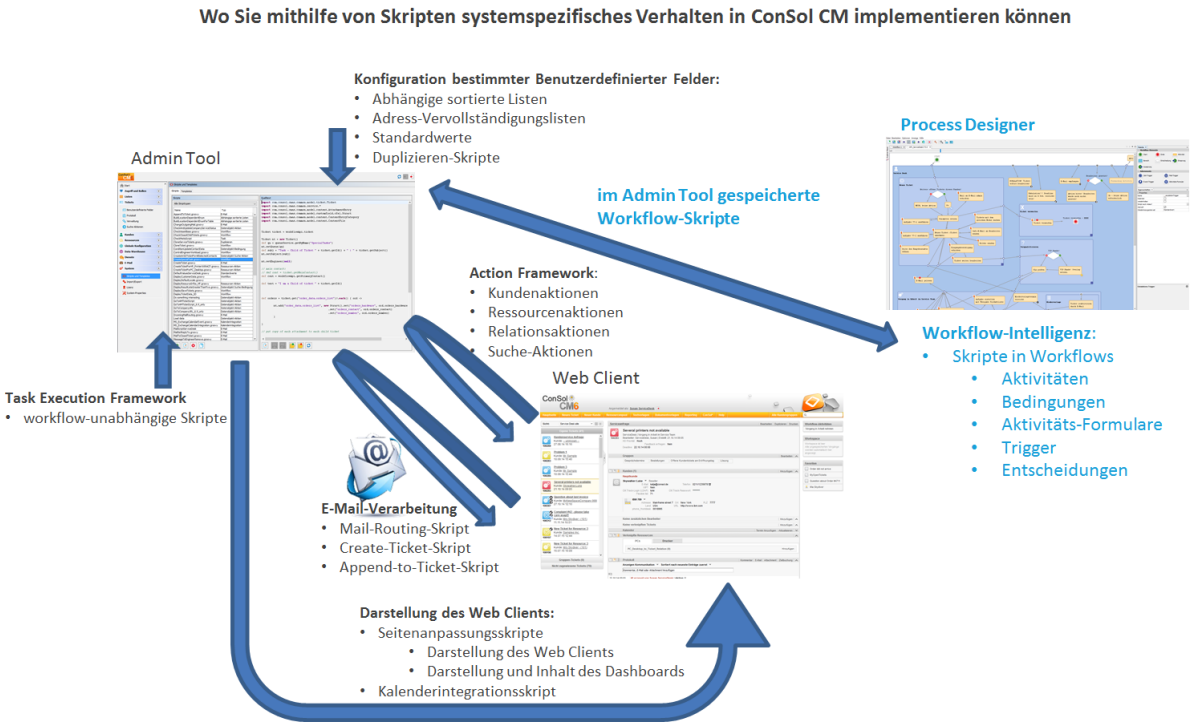


Abbildung 1: *Skripte für die Anpassung von ConSol CM und ihr Speicherort im System*

A.4 Rechtlicher Hinweis

Da wir ein Handbuch zur Verfügung stellen möchten, das Sie bei der Verwaltung Ihres CM-Systems unterstützt und Ihnen gleichzeitig zusätzliche Informationen über verwandte Themen bietet, enthält dieses Handbuch externe Links. Auf diese Weise können Sie bei Bedarf Hintergrundinformationen über ein Thema erhalten. Dies kann Ihnen dabei helfen, die benötigte CM-Konfiguration besser zu verstehen. Trotz sorgfältiger Prüfung übernehmen wir keine Haftung für den Inhalt dieser externen Links. Für ihren Inhalt sind ausschließlich die Betreiber der verlinkten Seiten verantwortlich.

A.5 Gender-Disclaimer

Soweit möglich sind ConSol CM Handbücher gender-neutral geschrieben und sprechen Sie als Leser oft mit "Sie" an. Wenn Formulierungen wie "Der Benutzer ..." verwendet werden, bezieht diese gewählte männliche Form immer gleichermaßen weibliche Personen ein. Auf konsequente Doppelbezeichnung wurde aufgrund der besseren Lesbarkeit verzichtet.

A.6 Copyright

A.7 Erklärungen zum Layout

Es werden folgende Symbole und Farben verwendet, um Informationen hervorzuheben bzw. zu markieren.

Dies ist eine zusätzliche Information.

Dies ist ein wichtiger Hinweis. An dieser Stelle müssen Sie besonders aufpassen!

Dies ist eine Warnung!

Dies ist eine Empfehlung aus der praktischen Erfahrung unserer Consultants.

A.8 Geschäftsprozesse

In einem Geschäftsprozess muss eine bestimmte Anzahl an Aufgaben in einer festgelegten Reihenfolge ausgeführt werden, um ein bestimmtes Ziel zu erreichen.

Die folgenden Komponenten sind (normalerweise) in Geschäftsprozessen relevant. Im Abschnitt Grundkomponenten von ConSol CM-Prozessen finden Sie eine Übersicht über die ConSol CM-Objekte, die diese Komponenten darstellen.

- **Prozess**
  Dies ist eine Sammlung von Aufgaben, die in einer bestimmten Reihenfolge ausgeführt werden. Aufgaben können nacheinander oder gleichzeitig ausgeführt werden. In ConSol CM wird der Prozess durch einen oder mehrere Workflows abgebildet. ConSol CM kann sowohl einzelne Prozesse als auch komplexe Prozessketten abbilden.
  Jeder Prozess muss eine definierte Eingabe und eine definierte Ausgabe haben. Das Objekt, das einen Vorgang darstellt und den Prozess durchläuft, heißt *Ticket*. Für den Endbenutzer kann es mit *Ticket* oder *Vorgang* oder einem anderen gewünschten Begriff bezeichnet werden.
- **Rollen und Verantwortlichkeiten**
  Normalerweise haben die Personen, die an einem Prozess beteiligt sind, unterschiedliche Rollen, d. h. unterschiedliche Verantwortlichkeiten. In ConSol CM kann jeder Bearbeiter, d. h. jede Person, die mit dem System arbeitet, eine oder mehrere Rollen haben.
- **Zugangsberechtigungen**
  Mit einem Business-Process-Management-System können unterschiedliche Prozesse in einer Firma gesteuert werden. Daher ist die Vergabe und Kontrolle von Zugangsberechtigungen eine Kernfunktionalität. In ConSol CM werden die Zugangsberechtigungen über Rollen zugewiesen.
- **Kunde**
  Dies ist die Person, die ein Interesse am Ergebnis des Prozesses hat. In ConSol CM gibt es immer einen Hauptkunden für ein Ticket. Dies kann eine Person, d. h. ein Kontakt oder eine Firma sein. Es können weitere Kunden hinzugefügt werden.
- **Aufgaben**
  Jede Aufgabe, die auch Vorgang, Anfrage oder ein anderer für den entsprechenden Anwendungsfall passender Begriff genannt werden kann, wird in ConSol CM als *Ticket* behandelt. In einem Geschäftsprozess können für ein Ticket verschiedene Arten von Aktivitäten durchgeführt werden:
  - manuelle Aktivitäten
  - systemgestützte Aktivitäten
  - vollständig automatisierte Aktivitäten

  In ConSol CM können alle Arten von Aufgaben verwaltet werden. Für manuelle Aufgaben gibt es To-Do-Listen für die Bearbeiter und mehrere Mechanismen, die sicherstellen, dass keine Aufgabe vergessen oder ignoriert wird.

A.9 Einführung in Workflows in ConSol CM

Eine der Kernkomponenten von ConSol CM ist die leistungsstarke Workflow-Engine. Daher wird ein Prozess in ConSol CM durch einen **Workflow** abgebildet. Das ist die technische Darstellung der aufeinanderfolgenden Schritte, die zur Erfüllung aller Aufgaben, die innerhalb des Geschäftsprozesses ausgeführt werden sollen, erforderlich sind.

Beispiele:

In einer IT-Helpdesk-Umgebung könnte ein Workflow aus den folgenden Schritten bestehen:
Neues Ticket - Ticket akzeptieren - An einer Lösung arbeiten - Den Kunden informieren - Ticket schließen.

In einem Sales-Prozess könnten die Schritte sein:
Erstkontakt: Lead - Zweitkontakt: Opportunity - Vertragskandidat - Vertrag.

Der Workflow, der alle benötigten Schritte enthält, wird in einer Workflow-Engine ausgeführt. In diesem Handbuch lernen Sie die Details über die Komponenten eines Workflows kennen und erfahren, wie man diese zur Entwicklung eines Workflows, der Ihren Geschäftsprozess abbildet, verwendet.

Ein Workflow...

- stellt einen bestimmten Prozess dar, z. B. die Schritte die zur Bearbeitung einer Kundenanfrage ausgeführt werden müssen.
- legt die Reihenfolge für die Aktivitäten und Entscheidungen fest.
- definiert die möglichen Pfade, die ein Ticket durchlaufen kann.

Der Vorgang oder die Anfrage, die bearbeitet werden soll, wird durch ein **Ticket** dargestellt, d. h. das Objekt, das den Workflow durchläuft.

Die folgende Abbildung zeigt eine grafische Darstellung eines einfachen Helpdesk-Prozesses.

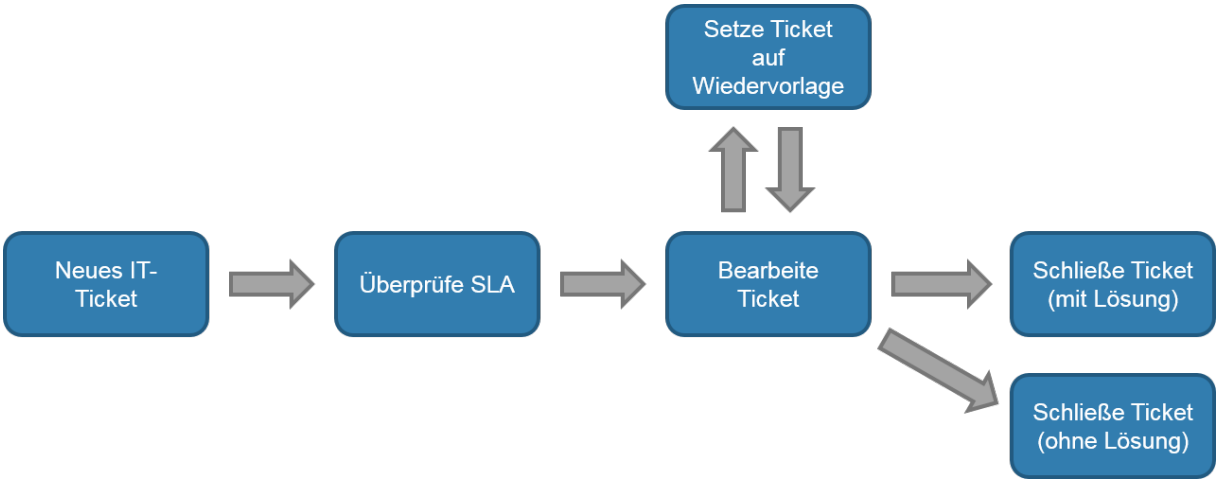


Abbildung 2: *Grafische Darstellung eines einfachen Geschäftsprozesses (IT-Helpdesk)*

A.10 Der ConSol CM Process Designer auf einen Blick

In diesem Kapitel werden folgende Themen behandelt:

A.10.1 Modellierung von Workflows

Ein Geschäftsprozess wird in ConSol CM mit dem *Process Designer* modelliert, einer Applikation, die ein integraler Bestandteil einer Standardinstallation von ConSol CM ist. Ein Prozess kann durch einen oder mehrere Workflows abgebildet werden, d. h. Sie verwenden den *Process Designer*, um Workflows zu entwickeln.

> In ConSol CM-Terminologie bezeichnet ein *Workflow* immer die technische Entität, während ein *Prozess* den Geschäftsprozess aus der logischen oder fachlichen Sicht bezeichnet.

Einer der Vorteile des Process Designers ist, dass es keine Lücke zwischen Workflow-Design und Workflow-Implementierung gibt. Mit der grafischen Benutzeroberfläche des Process Designers können Sie einen Workflow für einen Prozess entwickeln. Sobald Sie den Workflow einer Queue zugewiesen sowie die Rollen und Bearbeiter definiert haben, wird der Prozess lebendig und die Bearbeiter können damit arbeiten. Dies bedeutet, dass Sie den Process Designer für die beiden Schritte, die für die Erstellung von IT-gestützten Geschäftsprozessen wichtig sind, benutzen können:

- Modellierung und Entwicklung des Prozesses aus der logischen Perspektive
- Implementierung des Prozesses in einer technischen Instanz

Aufgrund dieser Flexibilität können Sie mit einer einfachen Version eines Workflows beginnen, üblicherweise in einer Test-Umgebung, und iterativ die gewünschten Funktionalitäten des Prozesses entwickeln. Das Team der Bearbeiter kann in jedem Schritt des Entwicklungs- und Optimierungsprozesses testen, ob die Anwendungsfälle wie gewünscht abgebildet sind.

Die grafische Darstellung eines Workflows im Process Designer ist der *Business Process Model and Notation* (BPMN) sehr ähnlich, sodass Sie intuitiv arbeiten können.

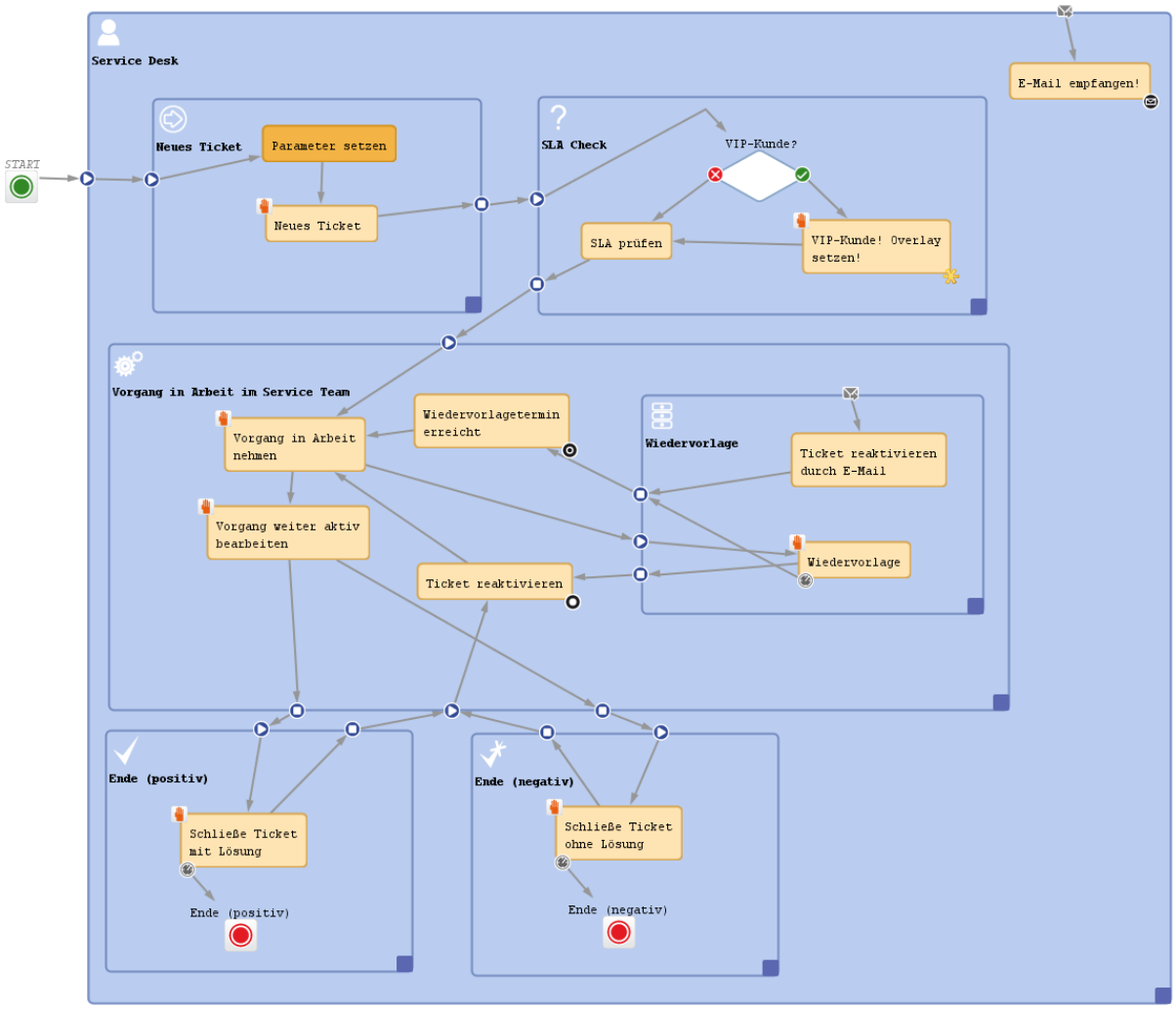


Abbildung 3: *ConSol CM Process Designer - Workflow-Modellierung, Beispielprozess*

Lesen Sie die folgenden Abschnitte, um einen ersten Eindruck von den Funktionen des Process Designers zu erhalten. Alle Themen werden in den entsprechenden Kapiteln dieses Handbuchs im Detail erklärt.

A.10.2 Tickets und Aktivitäten

Jeder Vorgang, der bearbeitet werden soll, ist ein *Ticket*. Dies bedeutet, dass ein Ticket einen konkreten Durchlauf durch den Workflow darstellt. Dies kann eine Anfrage, eine Bestellung oder eine andere Aufgabe sein, die in einem Geschäftsprozess bearbeitet werden soll.

Wenn ein neues Ticket in ConSol CM erstellt wird, wird es einem Workflow zugewiesen (mittels der Queue, zu der es gehört). Zuerst befindet sich das neue Ticket im START-Knoten. Im Laufe seines weiteren Lebenszyklus bewegt sich das Ticket durch die verschiedenen Aktivitäten des Workflows. Eine Aktivität ist die kleinste Einheit eines Workflows. Sie stellt einen einzelnen Schritt im Geschäftsprozess dar. Der Lebenszyklus eines Tickets endet, wenn es einen ENDE-Knoten erreicht.

Sie modellieren einen Prozess in einem Workflow, indem Sie Aktivitäten in einer bestimmten Reihenfolge verbinden. Das Ergebnis ist ein gerichteter Flussgraph. Er zeigt, welche Aktivitäten ausgeführt werden müssen, damit sich ein Ticket erfolgreich durch den Workflow (und damit den Geschäftsprozess) bewegt. Workflows können Zweige besitzen, sodass verschiedene Flusswege möglich sind. Auf diese Weise können Sie sicherstellen, dass ein Ticket z. B. zuerst akzeptiert werden muss, daraufhin muss das Problem gelöst werden und danach muss die Lösung dokumentiert werden. Nur dann kann das Ticket geschlossen werden.

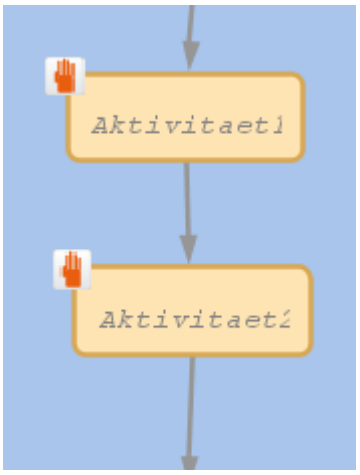


Abbildung 4: *ConSol CM Process Designer - Zwei aufeinanderfolgende manuelle Aktivitäten*

Es gibt manuelle und automatische Aktivitäten. Manuelle Aktivitäten erfordern eine Interaktion von einem Bearbeiter und werden als *Workflow-Aktivitäten* im Web Client angeboten. Automatische Aktivitäten werden hingegen ohne menschliche Eingabe ausgeführt und den Bearbeitern nicht angezeigt. Auf diese Weise kann ConSol CM dem Bearbeiter Zeit sparen und Daten aus verschiedenen Quellen hinter den Kulissen verarbeiten. Nur wenn eine Interaktion mit einem Bearbeiter nötig ist, wird der Prozess angehalten und wartet auf eine Eingabe durch den Bearbeiter.

Abbildung 5: *ConSol CM Web Client - Workflow-Aktivitäten*

A.10.3 Drag-and-Drop-Modellierung von Workflow-Komponenten

Sie können Ihren Workflow einfach und intuitiv mittels Drag-and-Drop modellieren. Ziehen Sie die benötigten Workflow-Elemente, z. B. eine Aktivität oder einen Entscheidungsknoten, aus der Palette in den Arbeitsbereich und verknüpfen Sie sie. Danach stellen Sie die Eigenschaften der Elemente im Eigenschaften-Editor ein.

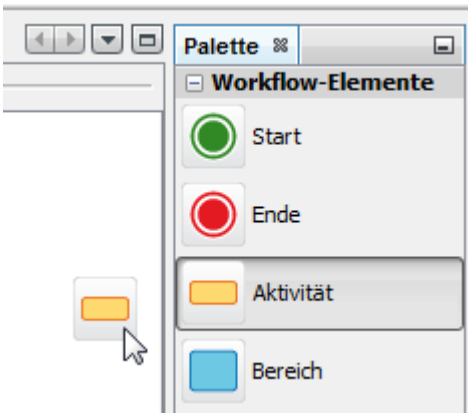


Abbildung 6: *ConSol CM Process Designer - Drag-and-Drop von Aktivitäten*

Mit den Grundelementen erstellen Sie Schritt für Schritt komplexe Workflows. Auf diese Weise können Sie selbst die kompliziertesten Geschäftsprozesse modellieren.

A.10.4 Bereiche und Verschachteln von Bereichen

Bei einem Prozess durchläuft ein Ticket verschiedene Status, z. B. Neues Ticket, Vorqualifikation, Aktive Arbeit und Dokumentation. Es kann auch sein, dass es für eine bestimmte Zeit auf Wiedervorlage gesetzt werden muss. Alle diese Status werden durch Bereiche (*Scopes*) dargestellt. In jedem Bereich können sich eine oder mehr Aktivitäten befinden. Dadurch ist es einfach, Workflows mit einer klaren Struktur zu entwickeln. Bereiche können auch hierarchisch organisiert werden, z. B. wenn ein Ticket während der Dokumentation auf Wiedervorlage gesetzt werden muss. Auf diese Weise können Sie mit hierarchischen Bereichen auch komplizierte Prozesse abbilden. Sie wählen die Detailtiefe dabei jederzeit selbst aus.

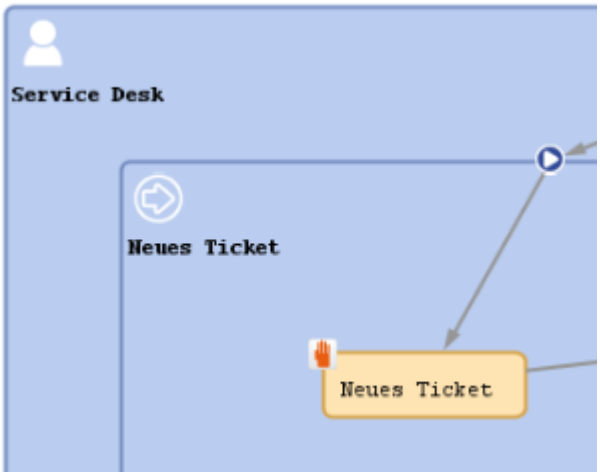


Abbildung 7: *ConSol CM Process Designer - Verschachteln von Bereichen*

A.10.5 Modellierung von Eskalationsmechanismen (Trigger und Wartezustände)

Bei den meisten Geschäftsprozessen ist das Einhalten von Terminen und Deadlines unentbehrlich. ConSol CM unterstützt durch automatische Zeit-Trigger die Einhaltung von Deadlines und bewahrt vor Verzögerungen. Diese Trigger messen z. B. die Reaktionszeit oder initialisieren Erinnerungsnachrichten.

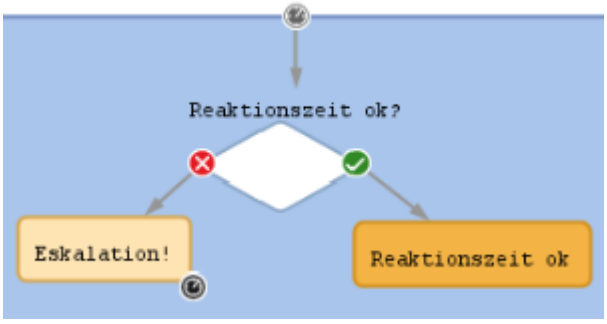


Abbildung 8: *ConSol CM Process Designer - Triggern von Aktionen*

A.10.6 Modellierung von Interrupts und Exceptions

Im echten Arbeitsalltag werden die Aufgaben eines Prozesses nicht immer Schritt für Schritt vollzogen, sondern können durch außergewöhnliche Ereignisse unterbrochen werden. Dies können verschiedene äußere Ereignisse sein. Solche Interrupts sequenziell zu modellieren, ist häufig sehr komplex oder sogar unmöglich. Der Process Designer stellt umfangreiche Tools zu diesem Zweck bereit.

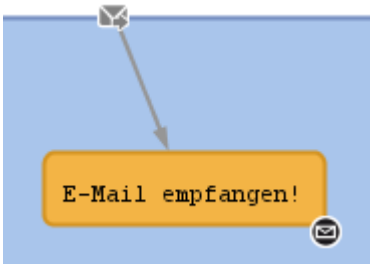


Abbildung 9: *ConSol CM Process Designer - Modellierung von Interrupts*

A.10.7 Skripte

Der Prozess, der als ConSol CM-Workflow modelliert wurde, besteht nicht nur aus Grundelementen wie Aktivitäten oder Entscheidungsknoten. Zu jedem Knoten eines Workflows kann ein Skript hinzugefügt werden, das die *Intelligenz* des Prozesses bereitstellt. Es können beispielsweise E-Mails an Kunden oder Bearbeiter verschickt, Interaktionen mit anderen Systemen implementiert oder Tickets übergeben werden. Grundsätzlich können alle Operationen, die mit Groovy-Skripten implementiert werden können, ausgeführt werden.

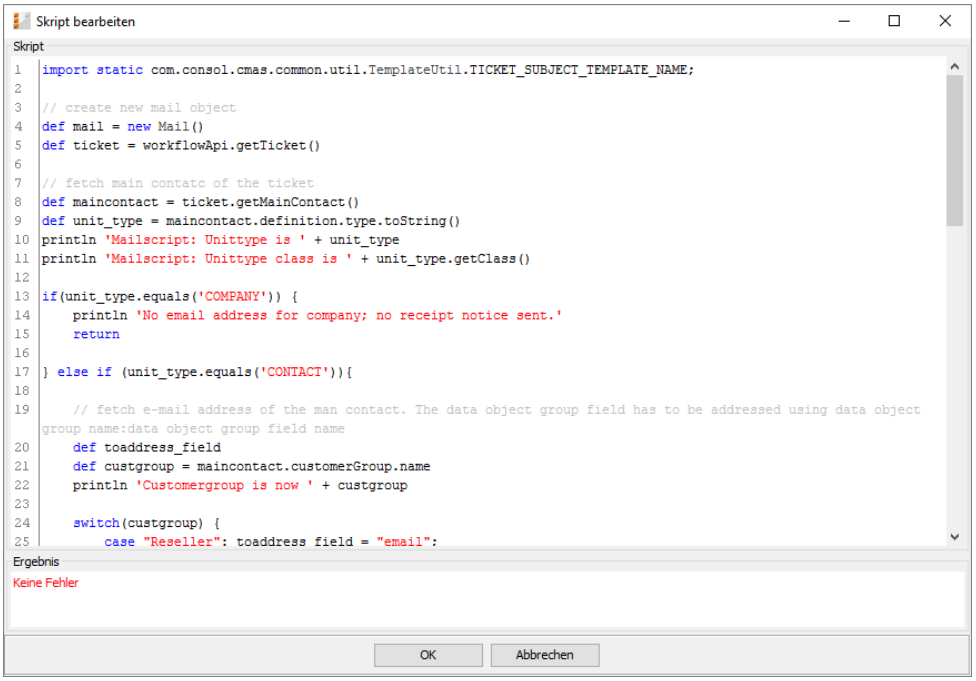


Abbildung 10: *ConSol CM Process Designer - Skript einer Aktivität*

A.10.8 Versionierung von Workflows

Geschäftsprozesse verändern sich ständig, z. B. aufgrund sich verändernder fachlicher und technischer Anforderungen. Der Process Designer verfügt über eine kontinuierliche Versionierung der installierten Workflows. Auf diese Weise können Sie einen neuen Workflow auf einfache Weise verwerfen (z. B. wenn Sie während der Systementwicklung eine neue Implementierung getestet haben) und zu einer der vorherigen Versionen zurückkehren.

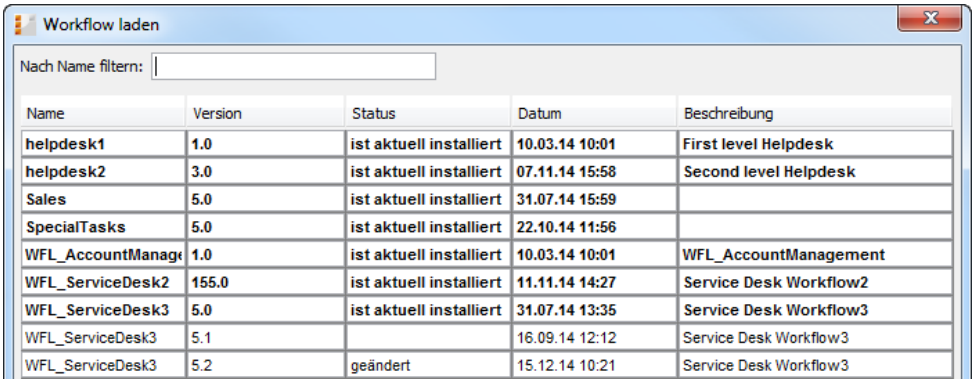


Abbildung 11: *ConSol CM Process Designer - Workflow-Versionen*

A.11 Grundkomponenten von ConSol CM-Prozessen

In diesem Kapitel werden folgende Themen behandelt:

A.11.1 Allgemeine Objekte

Während des Prozess-Designs und der Workflow-Entwicklung arbeiten Sie hauptsächlich mit den folgenden Objekten:

Notwendige Objekte:

- **Ticket**
  Das Ticket stellt den Vorgang dar. Abhängig vom Anwendungsfall kann dies ein Helpdesk-Vorgang, eine Sales-Opportunity, eine Bestellung oder eine Serviceanfrage sein.
- **Hauptkunde**
  Der Kunde ist die Person (Kontakt) oder Firma, die eine Frage oder eine Serviceanfrage hat. Diese Person oder Firma ist der Hauptkunde des Tickets. Sie stellt die externe Seite des CM-Systems dar.
- **Queue**
  Dies ist die organisatorische Einheit innerhalb des ConSol CM-Systems, die Tickets eines Fachbereichs zusammenfasst und die der zentrale Punkt für die Zuweisung von Zugangsberechtigungen und des Workflows ist. Eine Queue hat genau einen Workflow, der nicht geändert werden kann. Innerhalb einer Firma kann es beispielsweise eine Queue für die Sales-Abteilung, eine für den Customer Service und eine für die interne IT geben.
- **Bearbeiter**
  Dies ist die Person, die für die Erfüllung der Aufgaben in einem Ticket verantwortlich ist. Ein ConSol CM-Bearbeiter hat ein Login und ein Passwort für den Web Client. Der Hauptbearbeiter kann auch Ticketbesitzer genannt werden. Dieser kann sich im Verlauf des Prozesses ändern. Der Bearbeiter stellt die interne Seite des Systems dar.
- **Ressource**
  Dies stellt ein Objekt in dem Abschnitt der CM-Datenbank dar, mit dem CM.Resource Pool abgebildet ist. Das Objekt stellt je nach Design des Ressourcenpools ein Asset, einen Vertrag, eine Person, ein Produkt oder ein anderes Objekt dar. CM.Resource Pool ist ein ConSol CM-Add-on, das nicht zur CM-Standarddistribution gehört.
- **Workflow**
  Dies ist das Design oder Modell für den Prozess. Ein Workflow wird einer Queue zugewiesen (und kann mehr als einer Queue zugewiesen werden). Auf diese Weise durchlaufen alle Tickets, die sich in dieser Queue befinden, den Prozess, der durch diesen Workflow abgebildet wird. Die Workflow-Elemente, z. B. Aktivitäten, Bedingungen oder Entscheidungen, sind die wichtigsten

Mittel zur Konfiguration und Steuerung des Prozessflusses in ConSol CM. Ein Workflow kann einer oder mehreren Queues zugewiesen werden, z. B. können das IT-Servicedesk-Team und das Customer-Service-Team beide mit dem Workflow *serviceWorkflow* arbeiten.

- **Benutzerdefinierte Felder**
  Dies sind die Datenfelder, die zur Definition des Datenmodells für Ticketdaten verwendet werden. Sie legen auch das GUI-Design des Web Clients fest. Benutzerdefinierte Felder werden nie als Einzelfelder definiert, sondern immer in *Benutzerdefinierten Feldgruppen*.
- **Datenobjektgruppenfelder**
  Dies sind die Datenfelder, die zur Definition des Datenmodells für Kundendaten verwendet werden. Sie legen auch das GUI-Design des Web Clients fest.
  Datenobjektgruppenfelder werden nie als Einzelfelder definiert, sondern immer in *Datenobjektgruppen*.
- **Ressourcenfelder** (CM-Versionen 6.10 und höher, nur wenn CM.Resource Pool aktiv ist)
  Dies sind die Datenfelder, die zur Definition des Datenmodells für Ressourcendaten verwendet werden. Sie legen auch das GUI-Design des Web Clients fest.
  Ressourcenfelder werden nie als Einzelfelder definiert, sondern immer in *Ressourcenfeldgruppen*.

Optionale Objekte:

- **Ein oder mehrere Zusatzkunden**
  Neben dem Hauptkunden, d. h. dem Hauptkontakt oder (Version 6.9 oder höher) der Hauptfirma, können zusätzliche Kontakte (oder Firmen) zum Ticket hinzugefügt werden. Jedem Zusatzkunden kann eine Kundenrolle zugewiesen werden. Es könnte z. B. einen Stellvertreter für denjenigen, der das Ticket erstellt hat, geben oder der Teamleiter soll ebenfalls ein Kontakt für einen Support-Fall sein. Ein Zusatzkunde kann während des Prozesses der Hauptkunde werden und umgekehrt.
- **Ein oder mehrere zusätzliche Bearbeiter**
  Einem Ticket können zusätzliche Bearbeiter mit speziellen Rollen, die nach Bedarf definiert werden, zugewiesen werden. Zum Beispiel könnte ein Supervisor als zusätzlicher Bearbeiter gesetzt werden, um eine Genehmigung zu erteilen (Rolle *Genehmiger*), oder ein QA-Teammitglied kann mit der Rolle *QA* zum Ticket hinzugefügt werden, um das Ergebnis zu überprüfen, bevor das Ticket geschlossen wird.

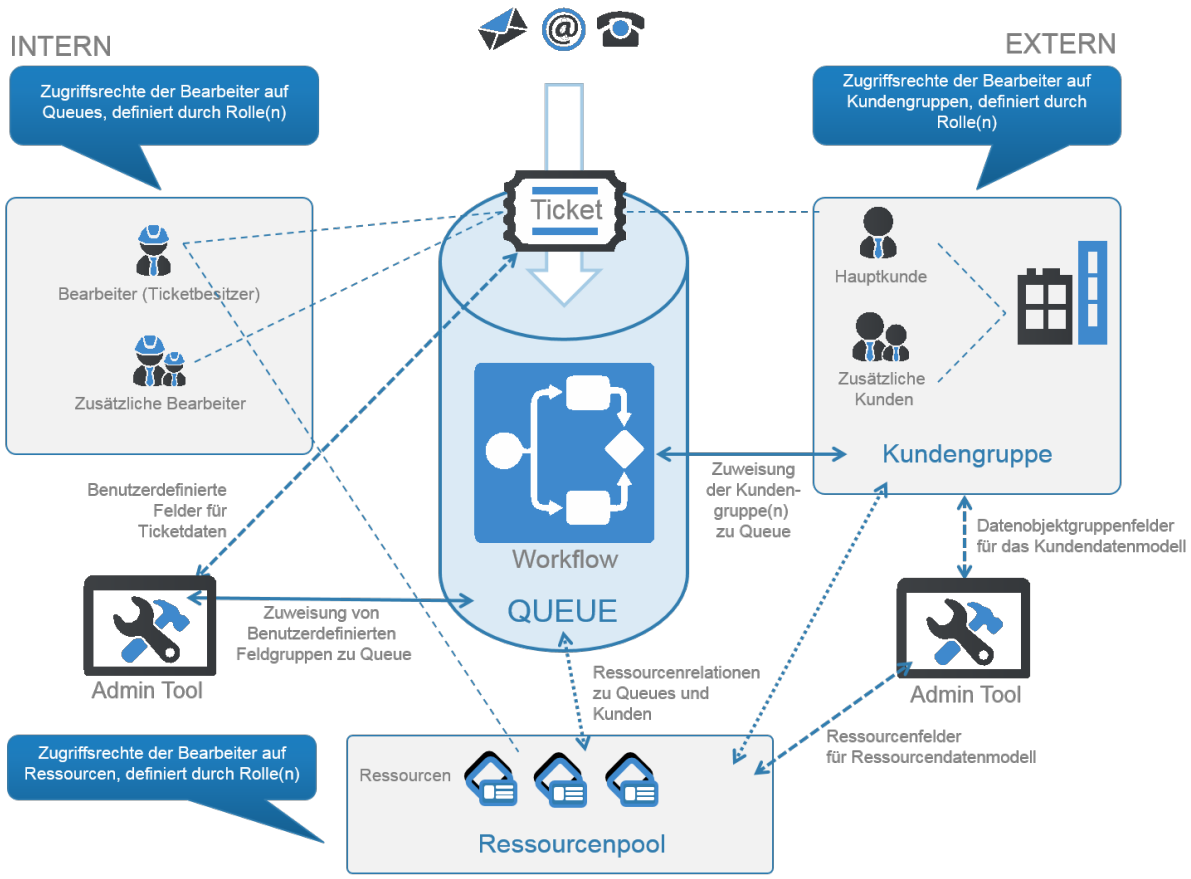


Abbildung 12: *ConSol CM - Grundprinzip*

A.11.2 Datenfelder

Eine detaillierte Erklärung zu diesem Thema finden Sie im Abschnitt Arbeiten mit Datenfeldern.

Es gibt drei Arten von Datenfeldern:

- **Benutzerdefinierte Felder**
  Werden zur Definition von Ticketdaten verwendet und in Benutzerdefinierten Feldgruppen verwaltet. Aus früheren CM-Versionen bekannt.
- **Datenobjektgruppenfelder**
  Werden zur Definition von Kundendaten in FlexCDM, dem neuen Kundendatenmodell, verwendet und in Datenobjektgruppen verwaltet.
- **Ressourcenfelder** (wenn CM.Resource Pool aktiv ist)
  Werden zur Definition von Ressourcendaten im Ressourcendatenmodell verwendet und in Ressourcenfeldgruppen verwaltet.

Sie können mit folgender Notation auf den Inhalt von einem Benutzerdefinierten Feld, Datenobjektgruppenfeld oder Ressourcenfeld zugreifen:

Ticket:

```
ticket.get("<group name>.<field name>")
```

Unit, für ein Feld:

```
unit.get("<group name>:<field name>")
```

Ressource:

```
resource.get("<group name>.<field name>")
```

Code-Beispiel 1: *Zugriff auf den Inhalt von Datenfeldern der drei CM-Hauptobjekte*

A.11.3 Standardfelder für Ticketdaten

Einige Felder müssen nicht als Benutzerdefinierte Felder im Admin Tool definiert werden, weil sie immer vorhanden sind. Das sind die folgenden Ticketfelder:

- **Ticket-ID**
  Wird dem Benutzer nicht angezeigt, nur interne Verwendung in der Datenbank.
- **Ticketname**
  Im Web Client sichtbar, wird normalerweise Ticketnummer genannt.
- **Ticketthema**
  Muss gesetzt werden.
- **Erstellungsdatum**
  Wird automatisch vom System gesetzt.
- **Bearbeiter**
  Kann Null oder einer der Bearbeiter sein.
- **Queue**
  Die aktuelle Queue des Tickets.

B - Arbeiten mit der Process-Designer-Applikation

In diesem Kapitel werden folgende Themen behandelt:

B.1 Benötigte Schritte für einen neuen Prozess

Die Arbeit mit dem Process Designer ist einer der ersten Schritte in der Abfolge von Schritten, die Sie durchführen müssen, wenn Sie einen neuen Prozess mit Bearbeitern, Rollen usw. erstellen möchten. Bevor wir Ihnen die Arbeit mit dem Process Designer erklären, erhalten Sie an dieser Stelle daher eine kurze Liste der Aufgaben, die zu erledigen sind:

1. Entwickeln und installieren Sie den Workflow mit dem Process Designer.
2. Erstellen Sie eine neue Queue mit diesem Workflow. Hier benötigen Sie außerdem die Definition aller erforderlichen Benutzerdefinierten Felder und Kundengruppen.
3. Erstellen Sie die Sichten für die neuen Bearbeiter anhand der Bereiche des neuen Workflows.
4. Erstellen Sie eine oder mehrere Rollen, die Zugriff auf die neue Queue haben. Beachten Sie dabei, dass der Zugriff auf die Kundengruppen dem Zugriff der Queue entsprechen muss.
5. Erstellen Sie einen oder mehrere Bearbeiter und weisen Sie diesen die neuen Rollen zu.
6. Überprüfen Sie die Anmeldung im Web Client. Können Sie in der neuen Rolle ein Ticket erstellen?

Alles, was Sie zum Durchführen dieser Schritte wissen müssen, ist detailliert im *ConSol Administratorhandbuch* beschrieben.

Beachten Sie, dass Sie, mit Ausnahme der Zuweisung eines Workflows, fast alle Parameter und Konfigurationen einer Queue fortlaufend verändern können. Sobald Sie der Queue allerdings einen Workflow zugewiesen haben, ist diese Zuweisung definitiv und kann nicht mehr geändert werden. Natürlich können Sie den Workflow selbst ändern, es ist aber nicht möglich, für diese Queue einen anderen Workflow auszuwählen. Das liegt daran, dass Änderungen an einem Workflow im Geschäftsprozess direkt nach der Installation wirksam werden und Widersprüche in jedem Fall vermieden werden müssen.

B.2 Starten des Process Designers

Sie können den Process Designer auf jedem PC oder Laptop starten, auf dem ein Standard-Webbrowser installiert ist (lesen Sie dazu bitte die *Systemanforderungen*) und der über das Netzwerk Zugriff auf den ConSol CM-Server hat.

Um den Process Designer zu starten, öffnen Sie die ConSol CM-Startseite und klicken auf den Link zum Process Designer. Für den Start der Process-Designer-Applikation ist Java Web Start (JWS) erforderlich, das auf dem lokalen Rechner läuft. Da JWS heute ein integraler Bestandteil aller Java-Distributionen ist, sollte dies kein Problem darstellen.

Falls der Process Designer nicht gestartet werden kann, könnte dies an der Netzwerkverbindung liegen. Überprüfen Sie die Java-Parameter für Netzwerkeinstellungen. Möglicherweise ist *Direkte Verbindung* erforderlich. Überprüfen Sie auch die Proxy-Einstellungen.

Auf einem Rechner sollte nur eine Instanz des Process Designers ausgeführt werden. Sie können die *javaw-Prozesse* auf dem Rechner kontrollieren, um sicherzugehen.

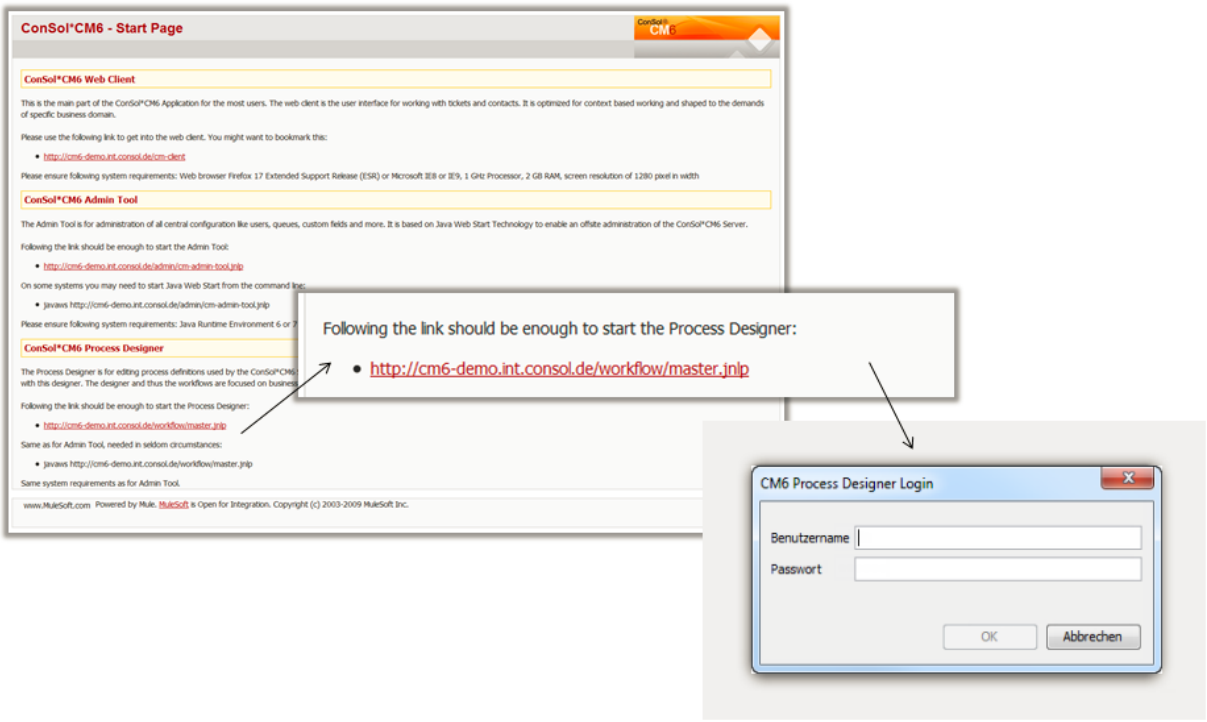


Abbildung 13: *ConSol CM - Starten des Process Designers*

Melden Sie sich mit einem Administratorkonto oder mit einem Konto mit Berechtigungen zur Workflow-Verwaltung an. Details dazu finden Sie im *ConSol CM Administratorhandbuch*, Abschnitt *Rollenverwaltung*.

B.3 Die GUI des Process Designers

In diesem Kapitel werden folgende Themen behandelt:

B.3.1 Einführung in die GUI-Elemente des Process Designers

B.3.1.1 Überblick: GUI-Bereiche

Die GUI des Process Designers enthält die folgenden Elemente, siehe folgende Abbildung und Liste:

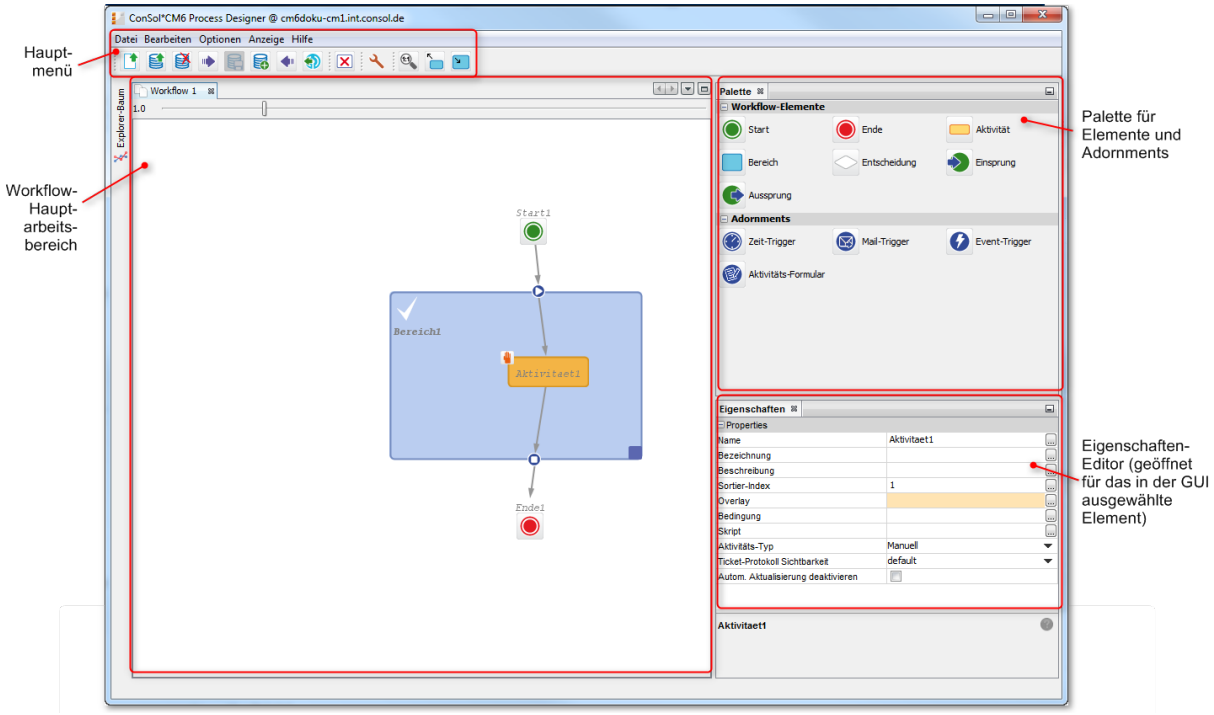


Abbildung 14: *ConSol CM Process Designer - GUI-Elemente*

B.3.1.2 Hauptmenü

Das Hauptmenü enthält die Menüpunkte als Texteinträge und eine Liste mit Menü-Icons.

Eintrag im Hauptmenü	Untereintrag	Icon	Bemerkung
Datei	Neu ...		Einen neuen Workflow erstellen.
	Laden ...		Einen Workflow laden. Öffnet eine Tabelle mit vorhandenen Workflows, siehe Abschnitt Laden eines Workflows.
	Löschen ...		Einen Workflow löschen. Öffnet eine Tabelle mit vorhandenen Workflows, siehe Abschnitt Löschen eines Workflows.

Eintrag im Hauptmenü	Untereintrag	Icon	Bemerkung
	Importieren ...		Importiert einen Workflow aus einer Datei (proprietäres Workflow-Format).
	Speichern ...		Workflow speichern (vorhandene Version).
	Speichern als neue Version		Den Workflow als neue Version speichern.
	Exportieren ...		Den Workflow in eine Datei exportieren. Öffnet den Dateibrowser des Betriebssystems. Der Workflow wird in einem proprietären Workflow-Format gespeichert (*.par*).
	Installieren ...		Workflow (als neue Version speichern) und installieren. Unter Umständen werden Sie zur folgenden Entscheidung aufgefordert: • Position im Prozess beibehalten (siehe Abschnitt Aktionen während der Workflow-Installation). • Prozess neu starten
	Login		Beim Process Designer anmelden. Normalerweise wird das Anmeldefenster direkt nach dem Start des Process Designers angezeigt. Zum Anmelden ist ein Konto mit Administratorberechtigungen oder mit den Berechtigungen zum Verwalten von Workflows (siehe *ConSol CM Administratorhandbuch*, Abschnitt *Rollenverwaltung*) erforderlich.
	Logout		Abmelden. Der Process Designer wird nicht beendet.
	Exit		Die Process-Designer-Applikation wird beendet/angehalten.
Bearbeiten	Alles aus aktuellem Reiter entfernen		Den gesamten Workflow mit allen Elementen aus dem Hauptbearbeitungsbereich löschen.

Eintrag im Hauptmenü	Untereintrag	Icon	Bemerkung
Optionen	Lokale Konfiguration		Pop-up-Fenster anzeigen, in dem Sie die Anzeigesprache des Process Designers auswählen können. Alle Sprachen, die für das System konfiguriert wurden (siehe Abschnitt *Globale Konfiguration* im *ConSol CM Administratorhandbuch*) sind verfügbar. Die Bezeichnungen werden im Workflow im Hauptbearbeitungsbereich in der ausgewählten Sprache angezeigt.
Anzeige	Normale Größe		Den Workflow in der Standardvergrößerung anzeigen (wie beim Start des Process Designers).
	Alle Bereiche ausklappen		Alle Bereiche (Scopes) in der ausgeklappten Version anzeigen. Siehe auch den folgenden Abschnitt Die GUI des Process Designers.
	Alle Bereiche zuklappen		Alle Bereiche (Scopes) in der eingeklappten Version anzeigen. Siehe auch den folgenden Abschnitt Die GUI des Process Designers.
	Palette ausblenden/einblenden		Palette auf der GUI ein-/ausblenden.
	Eigenschaften ausblenden/einblenden		Eigenschaften-Editor auf der GUI ein-/ausblenden.
	Explorer ausblenden/einblenden		Explorer-Baum ausblenden/einblenden.

Eintrag im Hauptmenü	Untereintrag	Icon	Bemerkung
	Zeige die Übertragungshistorie der Tickets an		Öffnet ein Pop-up-Fenster, in dem die Parameter für die Ticketübertragung während der Installation eines neuen Workflows angezeigt werden: • **Workflow-Name** Name des Workflows. • **Version** Version des alten Workflows. • **Startzeit** Start der Übertragung, dies ist die Startzeit der Operation *Installieren*. • **Endezeit** Ende der Übertragung, nach dieser Zeit ist der neue Workflow vollständig in Betrieb. • **Übertragene Tickets** Anzahl der Tickets, die übertragen wurden, d. h. die während der Workflow-Installation vom System angefasst wurden. Sollte identisch sein mit der Summe der offenen Tickets aus allen Queues, die diesen Workflow benutzen. • **Details** Zusätzliche Informationen bezüglich der Installation mit Ticketübertragung.
	IDE-Log		Öffnet den Log-Datei-Editor in der unteren Hälfte des Bildschirms und zeigt die benutzerspezifische Log-Datei des Process Designers an: *<USER_HOME_DIR>\.cmas\wfeditorR1\var\log*
Hilfe	About		Zeigt Versionsinformationen über den Process Designer und über die Java Virtual Machine, die in der aktuellen Konfiguration benutzt wird (dies ist die JVM des Browser-Plugins).

B.3.1.3 Workflow-Bearbeitungsbereich

Um einen Workflow zu entwickeln, definieren Sie die Workflow-Elemente im grafischen Layout-Modus des Process Designers und fügen, wenn benötigt, Skripte zu den Elementen hinzu.

Ein neues Element kann mittels Drag-and-Drop aus der Palette zum Workflow hinzugefügt werden.

Als Nachfolger eines vorhandenen Elements kann ein neues Element auch über das Kontextmenü (Rechtsklick) eines vorhandenen Elements erstellt werden, z. B. für eine Aktivität (siehe folgende Abbildung). Das neue Element und eine Verbindung zu diesem Element werden erstellt.

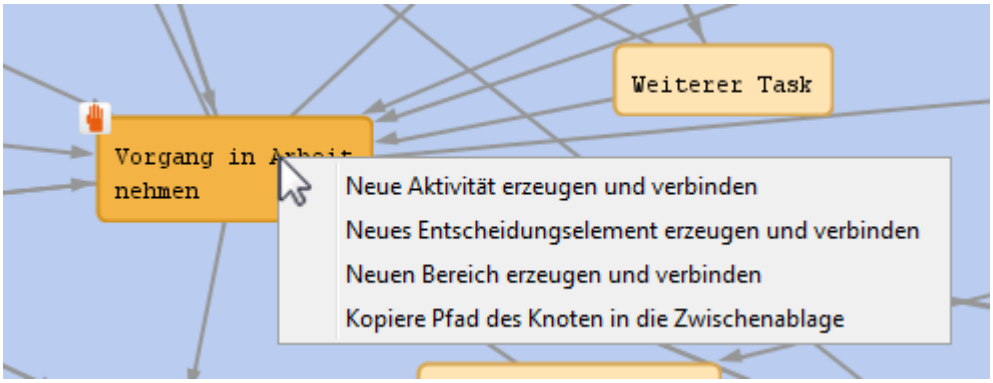


Abbildung 15: *ConSol CM Process Designer - Kontextmenü einer Workflow-Aktivität*

Eine neue Verbindung zwischen Elementen wird erstellt, indem man mit gedrückter linker Maustaste und gleichzeitig gedrückter *STRG-Taste* eine Linie zwischen den Elementen zieht. Wenn die Verbindung von einem Bereich in einen anderen Bereich führt, werden ein Eintritts- und Austrittspunkt automatisch hinzugefügt.

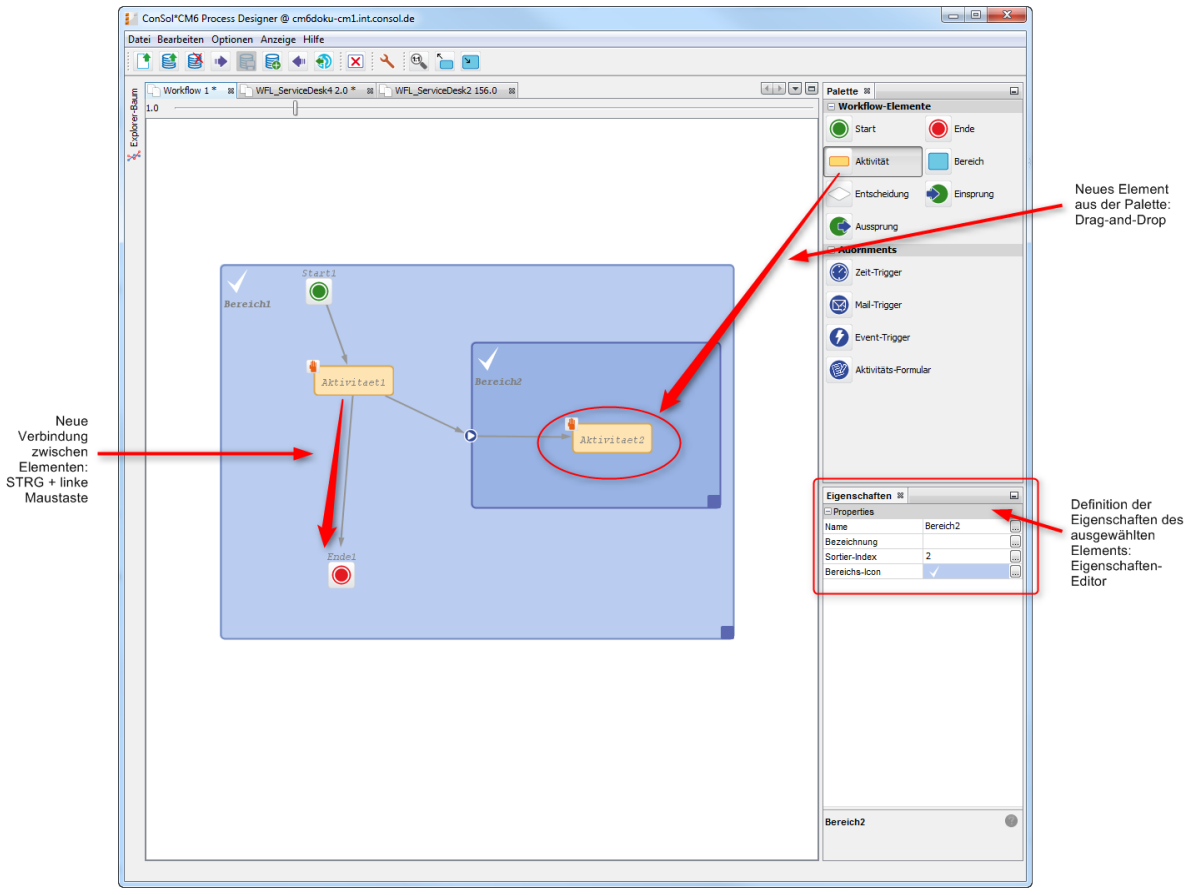


Abbildung 16: *ConSol CM Process Designer - Hinzufügen neuer Elemente und Verbindungen*

Es empfiehlt sich, für jeden Workflow einen globalen Bereich zu erstellen. Informationen über die Entwicklung von guten Workflows finden Sie im Abschnitt Best Practices.

Laden eines Workflows

Nachdem Sie das Icon oder den Menü-Eintrag *Laden* ausgewählt haben, wird Ihnen eine Tabelle mit allen verfügbaren Workflows angezeigt.

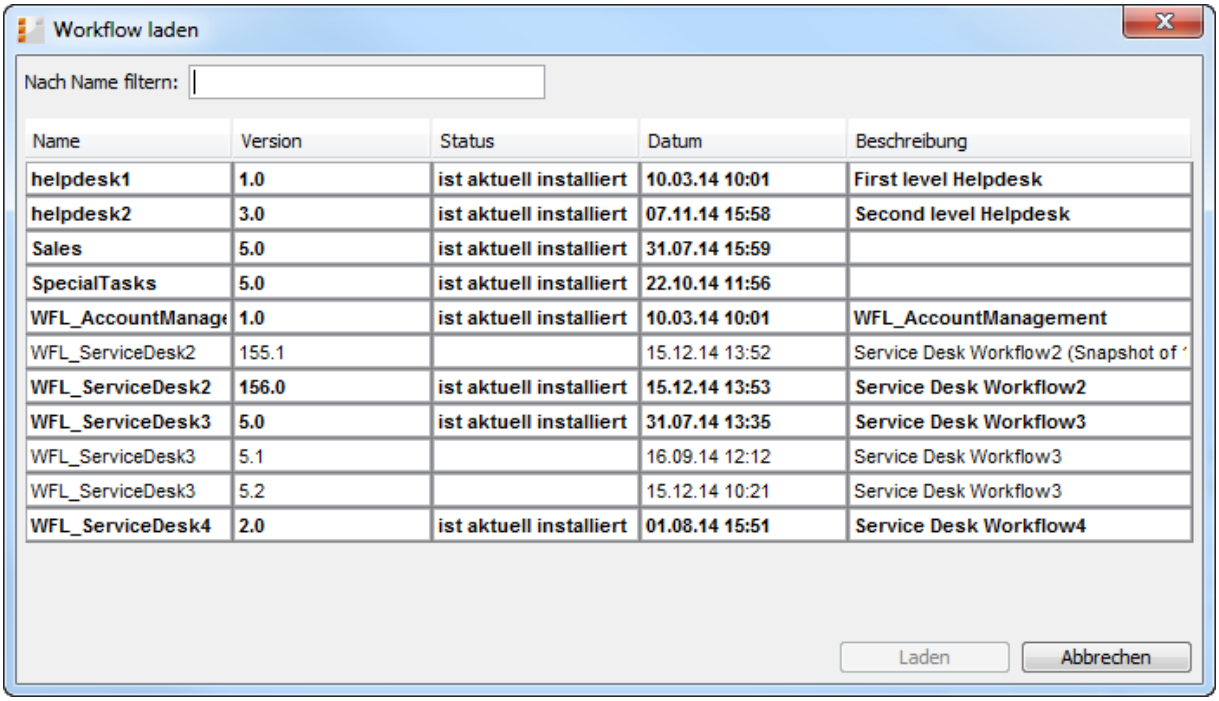


Abbildung 17: *ConSol CM Process Designer - Laden eines Workflows*

Sie können die Tabelle nach einer Spalte sortieren, indem Sie auf das kleine Dreieck neben der Spaltenüberschrift klicken.

Die Tabelle enthält die folgenden Spalten:

- **Name**
  Der Name des Workflows, der unter *Name* in den Eigenschaften des Workflows angegeben wurde (klicken Sie in den weißen Bereich, der sich um den globalen Bereich herum befindet, um die Eigenschaften des Workflows anzuzeigen).
- **Version**
  Die Version des Workflows. Diese wird vom ConSol CM-System automatisch zugewiesen. Wenn ein Szenario exportiert und danach wieder importiert wurde, beginnt die Nummerierung wieder mit 1.0.
- **Status**
  Bei älteren Workflows ist dieses Feld leer. Workflows, die installiert sind, werden mit *ist aktuell installiert* beschrieben.

- **Datum**
  Es wird das Datum der letzten Änderung (Datum, an dem der Workflow gespeichert wurde) angezeigt.
- **Beschreibung**
  Die Beschreibung, die im Feld *Workflow-Beschreibung* (**nicht** *Beschreibung*!) eingegeben wurde.

Um einen Workflow zu laden, wählen Sie ihn aus der Liste aus und klicken Sie auf *Laden*. Es kann nur ein Workflow ausgewählt werden.

Löschen eines Workflows

Nachdem Sie das Icon oder den Menü-Eintrag *Löschen* ausgewählt haben, wird Ihnen eine Tabelle mit allen verfügbaren Workflows angezeigt.

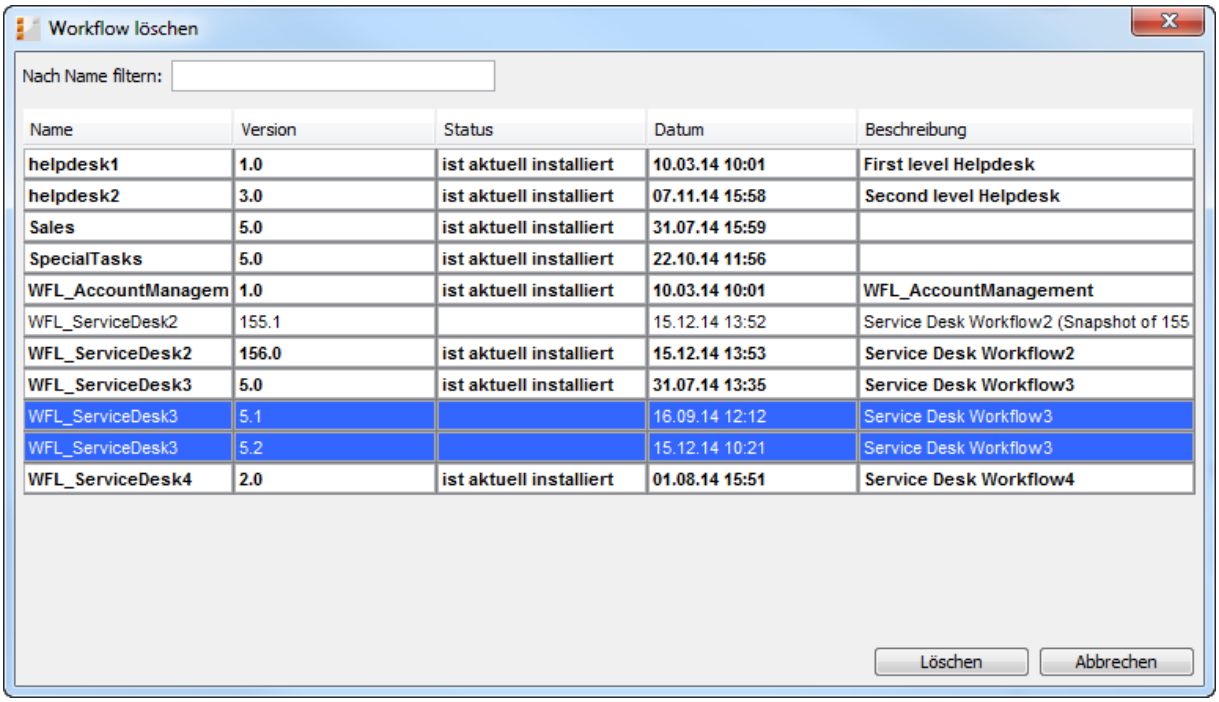


Abbildung 18: *ConSol CM Process Designer - Löschen von Workflows*

Sie können die Tabelle kann nach einer Spalte sortieren, indem Sie auf das kleine Dreieck neben der Spaltenüberschrift klicken.

Die Tabelle enthält die folgenden Spalten:

- **Name**
  Der Name des Workflows, der unter *Name* in den Eigenschaften des Workflows angegeben wurde (klicken Sie in den weißen Bereich, der sich um den globalen Bereich herum befindet, um die Eigenschaften des Workflows anzuzeigen).

- **Version**
  Die Version des Workflows. Diese wird vom ConSol CM-System automatisch zugewiesen. Wenn ein Szenario exportiert und danach wieder importiert wurde, beginnt die Nummerierung wieder mit 1.0.
- **Status**
  Bei älteren Workflows ist dieses Feld leer. Workflows, die installiert sind, werden mit *ist aktuell installiert* beschrieben.
- **Datum**
  Es wird das Datum der letzten Änderung (Datum, an dem der Workflow gespeichert wurde) angezeigt.
- **Beschreibung**
  Die Beschreibung, die im Feld *Workflow-Beschreibung* (**nicht** *Beschreibung*!) eingegeben wurde.

Um einen oder mehrere Workflow(s) zu löschen, wählen Sie die Workflows in der Liste aus und klicken Sie auf *Löschen*. Sie werden für jeden Workflow dazu aufgefordert, das Löschen des Workflows zu bestätigen. Wenn Sie also eine große Anzahl von Workflows zum Löschen ausgewählt haben und dann bemerken, dass Sie einen der Workflows doch nicht löschen möchten, können Sie dies tun, ohne den gesamten Vorgang abzubrechen.

Es empfiehlt sich, alle oder fast alle alten Workflows zu löschen, bevor Sie ein Szenario exportieren, da eine große Anzahl an Workflows die Größe eines Szenarios beträchtlich erhöht. Informationen über den Export und Import von Szenarios finden Sie im *ConSol CM Administratorhandbuch*.

B.3.1.4 Palette für Elemente und Adornments

Standardmäßig wird die Palette in der oberen rechten Ecke angezeigt. Sie können die Palette über den Eintrag im Hauptmenü *Palette ausblenden/einblenden* unter *Anzeige* aus- und wieder einblenden.

Die Palette enthält zwei Arten von Workflow-Komponenten:

- Elemente
- Adornments

Elemente

Elemente sind die grundlegende Komponenten, aus denen der Workflow besteht, und die die Prozesslogik abbilden.

Icon	Element	Bemerkung	Abschnitt
	Startknoten	Wird automatisch gesetzt, es kann kein weiterer Startknoten als der Standard-Startknoten hinzugefügt werden.	Workflow-Komponenten: Startknoten

Icon	Element	Bemerkung	Abschnitt
	Endknoten	Ein Workflow kann einen oder mehrere Endknoten haben.	Workflow-Komponenten: Endknoten
	Aktivität	Die Aktionen im Workflow, manuell oder automatisch.	Workflow-Komponenten: Aktivitäten
	Bereich (Scope)	Die höchste Hierarchieebene in Workflows	Workflow-Komponenten: Bereiche (Scopes)
	Entscheidung	Entscheidungsknoten, der einen Austrittspunkt *true* und einen Austrittspunkt *false* hat.	Workflow-Komponenten: Entscheidungsknoten
	Einsprung	Einsprungpunkt für Tickets aus anderen Workflows/Queues.	Aussprung- und Einsprungknoten
	Aussprung	Aussprungpunkt für Tickets. Es muss eine Ziel-Queue festgelegt werden. Ein Zielknoten kann definiert werden, ist aber optional.	Aussprung- und Einsprungknoten

Adornments

Adornments sind Objekte, die einer Workflow-Aktivität oder einem Bereich zugewiesen werden. Detaillierte Erklärungen finden Sie in den entsprechenden Abschnitten.

Icon	Adornment	Bemerkung	Abschnitt
	Zeit-Trigger	Kann Zeitintervalle messen und feuert, wenn das Ende des Intervalls erreicht wurde. Optional kann ein Arbeitszeitkalender verwendet werden.	Zeit-Trigger
	Mail-Trigger	Feuert, wenn eine E-Mail für das Ticket eingegangen ist.	Mail-Trigger
	Event-Trigger	Feuert, wenn ein Event eingetreten ist. Die Art des Events kann festgelegt werden (z. B. Änderung des Bearbeiters, Änderung der Priorität).	Event-Trigger
	Aktivitäts-Formular (Activity Control Form, ACF)	Legt fest, welches ACF angezeigt werden soll, wenn die Aktivität ausgeführt wird. Die ACFs werden im Admin Tool definiert.	Aktivitäts-Formulare (ACFs)

Arten von Aktivitäten

Aktivitäten können zusätzliche Icons haben, die angeben, ob die Aktivität manuell ist oder es Bedingungen gibt. Detaillierte Erklärungen finden Sie in den entsprechenden Abschnitten.

Icon	Name	Beschreibung	Abschnitt
	Manuell	Zeigt an, dass es sich bei der Aktivität um eine manuelle Aktivität handelt, d. h. der Bearbeiter muss im Web Client darauf klicken.	Aktivitäts-Typ
	Bedingung	Zeigt an, dass an der Aktivität ein Bedingungsskript hängt. Das Bedingungsskript wird ausgeführt, wenn die vorherige Aktivität durchgeführt wurde. Die Aktivität mit dem Bedingungs-Icon wird nur angezeigt, wenn das Bedingungsskript *true* zurückgibt.	Bedingung

B.3.1.5 Der Eigenschaften-Editor (Beispiel: Aktivität)

Der Eigenschaften-Editor ist für das Element geöffnet, das im Hauptarbeitsbereich ausgewählt wurde. Er enthält komponentenspezifische Parameter. Einige allgemeine Parameter sind für alle Komponenten sichtbar, manche sind nur für einen bestimmten Komponententyp vorhanden.

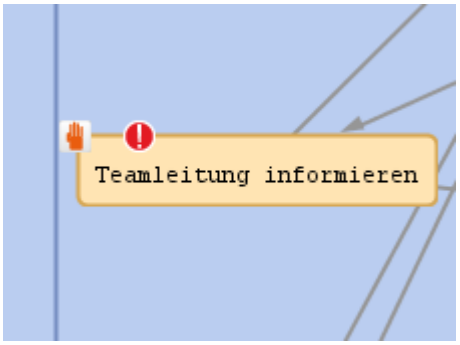


Abbildung 19: *ConSol CM Process Designer - Ausgewählte Aktivität im Workflow*

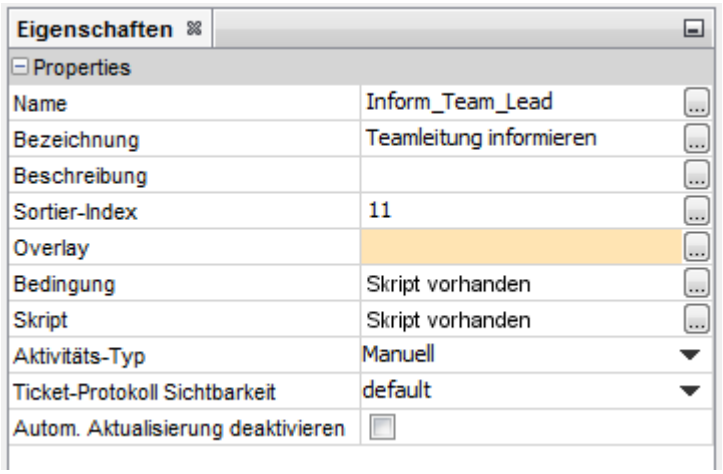


Abbildung 20: *ConSol CM Process Designer - Eigenschaften-Editor*

Eigenschaften:

- **Name**
  Pflichtangabe. Dies ist der technische Name des Objekts. Wenn ein Objekt neu erstellt wird,

können Sie seine Bezeichnung editieren und der Objektname wird automatisch aus der Bezeichnung erstellt (Umlaute werden ausgelassen). Danach wird der Name des Objekts nicht mehr automatisch geändert, kann aber noch manuell editiert werden. Zulässige Zeichen für Namen sind:

- Buchstaben (Klein- und Großbuchstaben), aber keine Umlaute
- Unterstriche
- Zahlen

- **Bezeichnung**
  Der lokalisierte Name des Elements. Alle Sprachen, die für das System konfiguriert wurden, sind verfügbar und können ausgefüllt werden. Im Web Client des Bearbeiters wird die Beschreibung in der Sprache angezeigt, die er im Webbrowser eingestellt hat. Wenn keine Bezeichnung in dieser Sprache verfügbar ist, wird die Bezeichnung in der Standardsprache angezeigt.

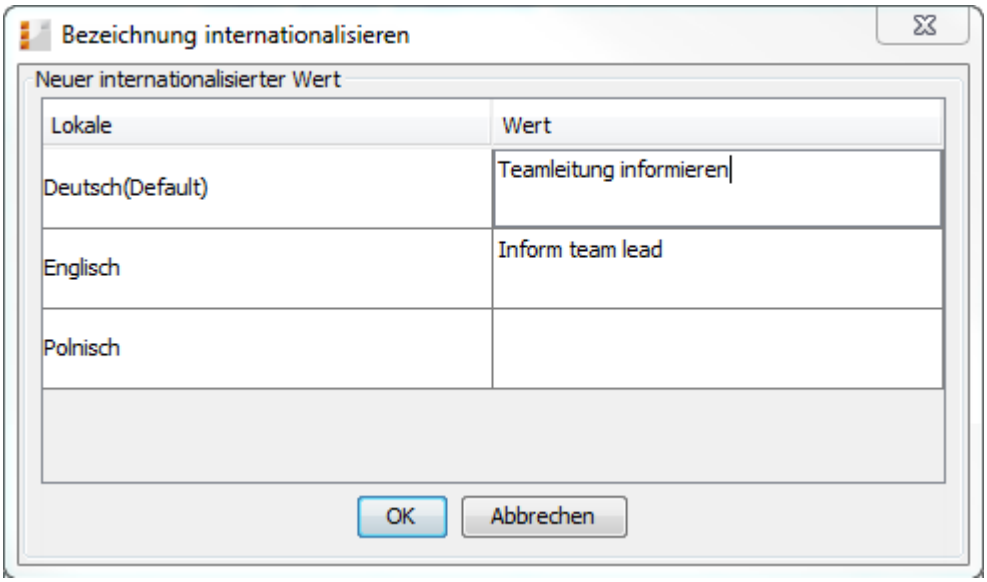


Abbildung 21: *ConSol CM Process Designer - Lokalisierung für Bezeichnungen*

- **Beschreibung**
  Optional. Es kann ein lokalisierter Text eingegeben werden, der als Mouseover im Web Client angezeigt wird. Dieser kann dem Bearbeiter Informationen dazu liefern, was passiert, wenn er die betreffende Workflow-Aktivität ausführt.

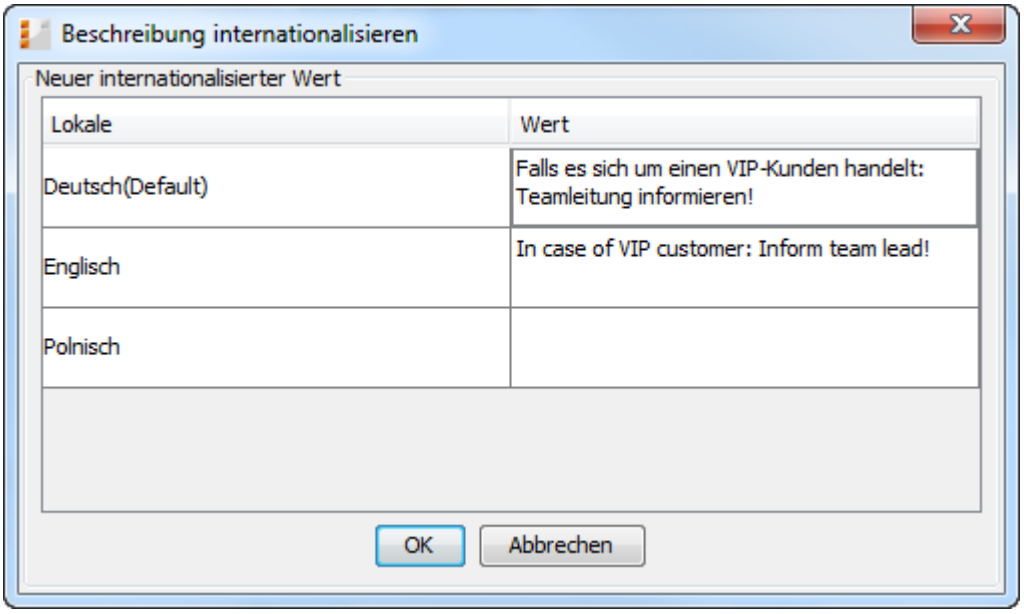


Abbildung 22: *ConSol CM Process Designer - Lokalisierte Beschreibung einer Aktivität*

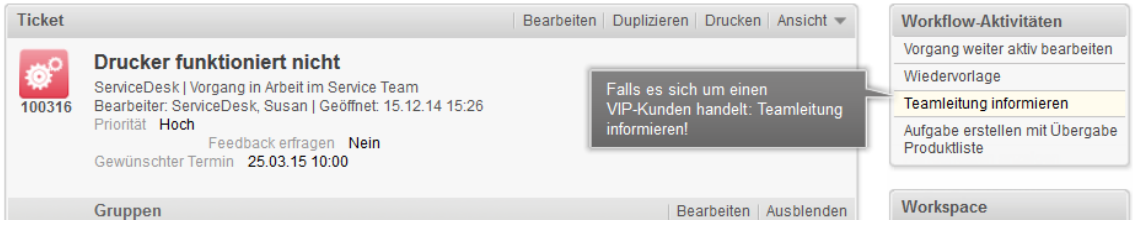


Abbildung 23: *ConSol CM Web Client - Lokalisierte Beschreibung einer Aktivität als Mouseover*

- **Sortier-Index**
  Definiert:
  - **Für Aktivitäten:**
    Die Reihenfolge der Aktivitäten in der Liste der *Workflow-Aktivitäten* im Web Client. Je höher die Zahl ist, desto weiter unten steht die Aktivität in der Liste im Web Client.
  - **Für Bereiche:**
    Die Reihenfolge der Tickets in der Ticketliste (Web Client) in Sichten. Je höher der Sortier-Index des Bereichs ist, desto weiter unten in der Ticketliste stehen die Tickets.
- **Overlay**
  Optional, für Aktivitäten. Klicken Sie in das orangene Feld, um ein Standard-Overlay von ConSol CM oder ein bereits hochgeladenes Overlay auszuwählen. Klicken Sie auf den Dateiexplorer-Button (...), um den Dateiexplorer des Betriebssystems zu öffnen und ein neues Icon hochzuladen. Wenn das Ticket die Aktivität durchläuft, wird das Overlay im Web Client zum Ticket-Icon hinzugefügt. Maximal können drei Overlays am Ticket-Icon hängen. Dieser Mechanismus kann für mehrere Zwecke verwendet werden. Einige Beispiele sind:

- **Eine Eskalation:**
  Das Ticket wurde erstellt und es hat sich noch kein Bearbeiter darum gekümmert.
- **Eine E-Mail:**
  Das Ticket hat eine E-Mail erhalten.
- **Eine Nachricht an den Bearbeiter:**
  Ein anderer Bearbeiter hat zum Beispiel einen Kommentar zu *meinem* Ticket hinzugefügt.

Abbildung 24: *ConSol CM Process Designer - Eigenschaften-Editor: Standard-Overlays und ein kundenspezifisches Overlay*

Abbildung 25: *ConSol CM Web Client - Ticket-Icons mit Overlays*

- **Overlay-Gültigkeit**
  Wird nur angezeigt, wenn ein Overlay ausgewählt wurde.
  - **Aktivität**
    Das Overlay hängt so lange am Ticket-Icon, wie das Ticket hinter der Aktivität steht. Sobald die nächste Aktivität ausgeführt wird, wird das Overlay vom Ticket-Icon entfernt.
  - **Bereich**
    Das Overlay wird gelöscht, wenn das Ticket den Bereich verlässt.
  - **Prozess**
    Das einmal zum Ticket-Icon hinzugefügte Overlay hängt für den Rest des Prozesses am Ticket-Icon.
  - **Nächstes Overlay**
    Das Overlay hängt so lange am Ticket-Icon, wie kein neues Overlay hinzugefügt wird. Wenn ein neues Overlay hinzugefügt wird, wird das alte Overlay gelöscht.
- **Bedingung**
  Optional, für Aktivitäten. Im Skripteditor kann ein Skript eingegeben werden (siehe Abschnitt Der Skripteditor), das *true* oder *false* zurückgeben muss. Das Skript wird ausgeführt, wenn die vorherige Aktivität durchgeführt wurde, d. h. sobald es möglich ist, die Aktivität mit der Bedingung anzuzeigen. Wenn *true* zurückgegeben wird, wird die Aktivität angezeigt. Wenn *false* zurückgegeben wird, wird die Aktivität nicht angezeigt. Eine Aktivität mit einer Bedingung hat das Icon *Ausrufezeichen*.

- **Skript**
  Optional, für Aktivitäten. Im Skripteditor kann ein Skript eingegeben werden (siehe Abschnitt Der Skripteditor), das ausgeführt wird, wenn das Ticket die Aktivität durchläuft.
- **Aktivitäts-Typ**
  Pflichtangabe, für Aktivitäten. Es muss *Manuell* oder *Automatisch* ausgewählt werden. Eine manuelle Aktivität wird im Web Client angezeigt und muss explizit von einem Bearbeiter ausgewählt/ausgeführt werden. Im Process Designer ist diese durch das Icon *Hand* gekennzeichnet.
  Eine automatische Aktivität wird ohne Beteiligung des Bearbeiters ausgeführt. Eine detaillierte Erklärung der Prozesslogik von ConSol CM finden Sie im Abschnitt Prozesslogik.
- **Ticket-Protokoll Sichtbarkeit**
  Pflichtangabe, ein Standardwert ist bereits gesetzt (*default*). Der Wert definiert das Sichtbarkeitslevel auf der GUI des Web Clients, auf dem die Aktion (Durchführung der Aktivität) angezeigt werden soll:
  - 2nd level and 3rd level
  - only 3rd level
  - on every level
  - default
    Dies bezieht sich auf den im Admin Tool unter *Ticketprotokoll* für die Aktivitätskonfiguration definierten Wert. Je nach Aktivitätstyp wird einer der folgenden Parameter verwendet:
    - Manuelle Aktivität oder Aktivität mit Overlay ausgeführt
    - Aktivität nach Eskalation ausgeführt
    - Automatische Aktivität ausgeführt

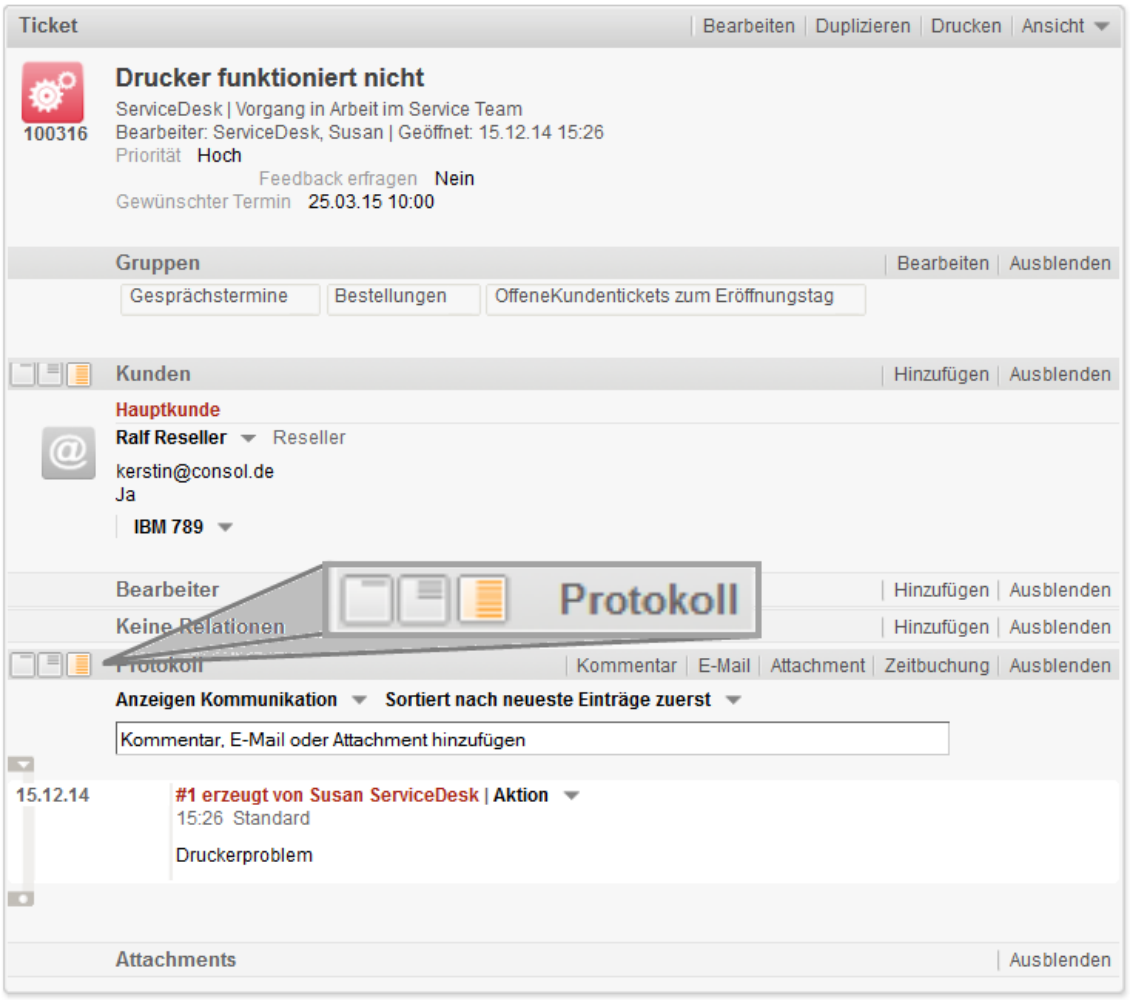


Abbildung 26: *ConSol CM Web Client - Sichtbarkeitslevel im Ticketprotokoll*

- **Autom. Aktualisierung deaktivieren**
  Legt das Verhalten des Tickets fest, wenn ein Ereignis gefeuert oder ausgeführt wurde. Normalerweise wird nach einem Ereignis automatisch eine Ticket-Update-Operation durchgeführt. Bei mehreren direkt aufeinanderfolgenden Ereignissen, sollten Sie es vermeiden, nach jedem einzelnen Ereignis eine Ticket-Update-Operation anzustoßen. Um dies zu vermeiden, setzen Sie für alle Ereignisse außer dem letzten Ereignis der Kette das Feld *Autom. Aktualisierung deaktivieren* auf *true*. So wird das Ticket erst nach dem letzten Ereignis aktualisiert.

B.3.2 Der Skripteditor

Mit dem Skripteditor schreiben Sie im Process Designer Groovy-Skripte (d. h. es wird reiner Groovy-Code und Java-Code akzeptiert). Erklärungen, Empfehlungen und Beispiele bezüglich der Workflow-Programmierung mit Skripten finden Sie im Abschnitt Einführung in die Workflow-Programmierung.

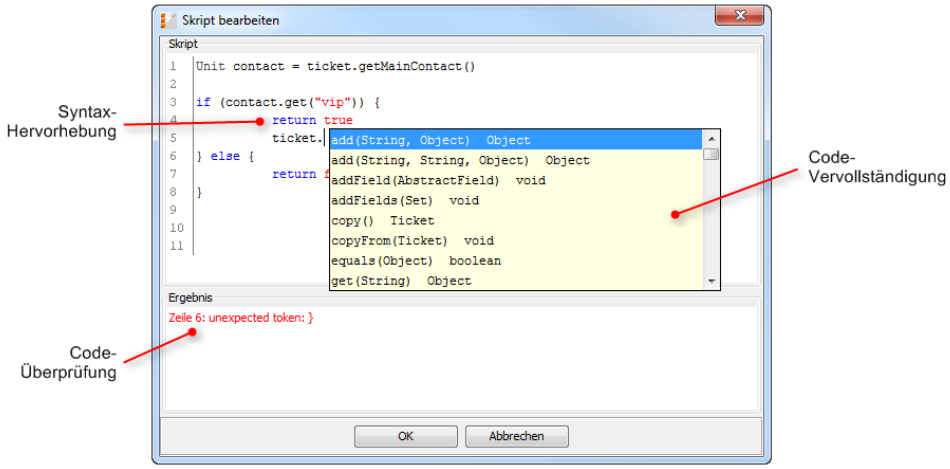


Abbildung 27: *ConSol CM Process Designer - Skripteditor*

Der Skripteditor hat folgende Funktionen:

- **Syntax-Hervorhebung**
  Groovy-Code wird entsprechend der Schlüsselwörter hervorgehoben.
- **Code-Vervollständigung**
  Wenn Sie den Namen eines Objekts und den Punkt eingegeben haben, werden mögliche Methoden vorgeschlagen. Drücken Sie *STRG* + *LEERZEICHEN*, um die Code-Vervollständigung zu aktivieren.
- **Code-Überprüfung**
  Der eingegebene Code wird auf den korrekten Gebrauch der allgemeinen Syntax und der Methoden hin geprüft. Der fehlerhafte Code wird im Bereich *Ergebnis* angezeigt.

B.3.3 GUI-Tipps und -Tricks

Denken Sie daran, den Workflow regelmäßig zu speichern, während Sie an ihm arbeiten. Es gibt momentan keinen Button zum Rückgängig machen. Falls Sie versehentlich Elemente entfernt haben, die Sie noch brauchen, laden Sie den Vorgänger des aktuellen Workflows und arbeiten Sie damit weiter.

Zum ...	führen Sie folgende Aktion aus ...
Ausklappen eines einzelnen eingeklappten Bereichs	Doppelklicken Sie in den Bereich.
Einklappen eines einzelnen ausgeklappten Bereichs	Doppelklicken Sie in den Bereich.
Ändern der Größe eines Bereichs	Ziehen Sie das kleine dunkelblaue Quadrat, das sich unten rechts am Bereich befindet.
Hinzufügen eines neuen Mail-Triggers zum Bereich	Klicken Sie in der Palette auf das Mail-Trigger-Symbol und ziehen Sie den neu erstellten Mail-Trigger in den entsprechenden Bereich.
Hinzufügen eines neuen Zeit-Triggers zum Bereich	Klicken Sie in der Palette auf das Zeit-Trigger-Symbol und ziehen Sie den neu erstellten Zeit-Trigger in den entsprechenden Bereich.
Hinzufügen eines neuen Zeit-Triggers zu einer Aktivität	Klicken Sie in der Palette auf das Zeit-Trigger-Symbol und ziehen Sie den neu erstellten Zeit-Trigger in die entsprechende Aktivität (z. B. für eine Wiedervorlage).
Löschen einer Verbindung zwischen zwei Aktivitäten	Markieren Sie die Verbindung (nicht den Bereich oder die umliegenden Elemente) und drücken Sie die Entfernen-Taste.
Hinzufügen einer neuen Kante in eine Verbindung zwischen zwei Aktivitäten	Klicken Sie auf die Verbindung: ein kleines Quadrat zeigt die neue Kante an, die verschoben werden kann.

C - Komponenten von ConSol CM-Workflows

C.1 Einleitung

Sie können mit verschiedenen Arten von Komponenten arbeiten, um die Workflows in Ihrem ConSol CM-System zu erstellen. Die Palette im Process Designer enthält alle Elemente und Adornments. Einen Überblick finden Sie in Palette für Elemente und Adornments.

In den folgenden Kapiteln werden alle Elemente und Adornments detailliert beschrieben.

Workflow-Element	Erklärung
Startknoten	Der erste Knoten in einem Workflow, siehe Abschnitt Workflow-Komponenten: Startknoten.
Endknoten	Ein oder mehrere Endknoten des Prozesses. Das Ticket ist geschlossen. Siehe Abschnitt Workflow-Komponenten: Endknoten.
Bereiche (Scopes)	Bereiche eines Prozesses, siehe Abschnitt Workflow-Komponenten: Bereiche (Scopes).
Aktivitäten	Die Schritte eines Prozesses. Können automatisch oder manuell sein, siehe Abschnitt Workflow-Komponenten: Aktivitäten.
Entscheidungsknoten	Workflow-Element, das eine Entscheidung *true/false* abbildet, siehe Abschnitt Workflow-Komponenten: Entscheidungsknoten.
Adornments	Elemente zur Steuerung des Prozessflusses: Trigger und Aktivitäts-Formulare. Siehe Abschnitt Adornments (Trigger und ACFs) .
Aussprung- und Einsprungknoten	Elemente, die Workflows verbinden, siehe Abschnitt Aussprung- und Einsprungknoten.

C.2 Workflow-Komponenten: Startknoten

Jeder Workflow enthält genau einen Startknoten. Wenn Sie einen neuen Workflow erstellen, wird der Startknoten automatisch hinzugefügt. Sie müssen ihn also nicht selber hinzufügen.

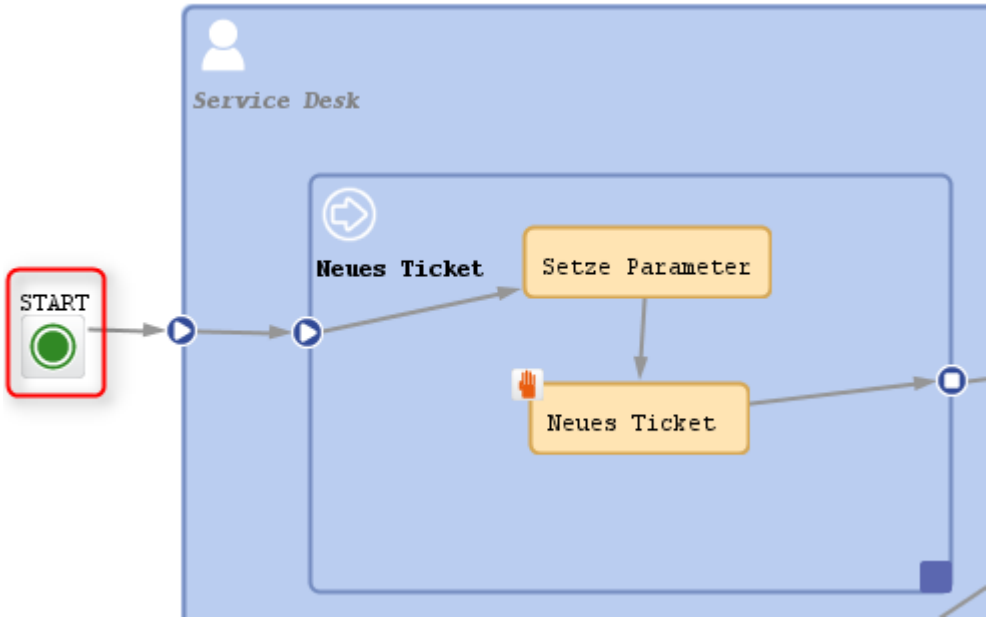


Abbildung 28: *ConSol CM Process Designer - Startknoten*

Der Startknoten hat keine Skripte und kann nicht konfiguriert werden.

Wenn ein Ticket den Workflow beginnt und kein spezieller Einstiegspunkt definiert wurde, durchläuft das Ticket den Startknoten.

Der Startknoten sollte nicht innerhalb des globalen Bereichs positioniert werden. Siehe auch Abschnitt Best Practices.

C.2.1 Eigenschaften eines Startknotens

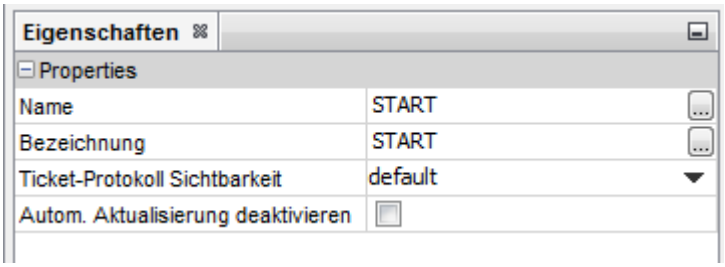


Abbildung 29: *ConSol CM Process Designer - Eigenschaften eines Startknotens*

Eigenschaften:

- **Name**
  Technischer Name des Objekts
- **Bezeichnung**
  Lokalisierter Name, der auf der GUI angezeigt wird.
- **Ticket-Protokoll Sichtbarkeit**
  Siehe Abschnitt Ticket-Protokoll Sichtbarkeit.
- **Autom. Aktualisierung deaktivieren**
  Siehe Abschnitt Autom. Aktualisierung deaktivieren.

C.3 Workflow-Komponenten: Endknoten

Ein Workflow in ConSol CM kann einen oder mehrere Endknoten haben.

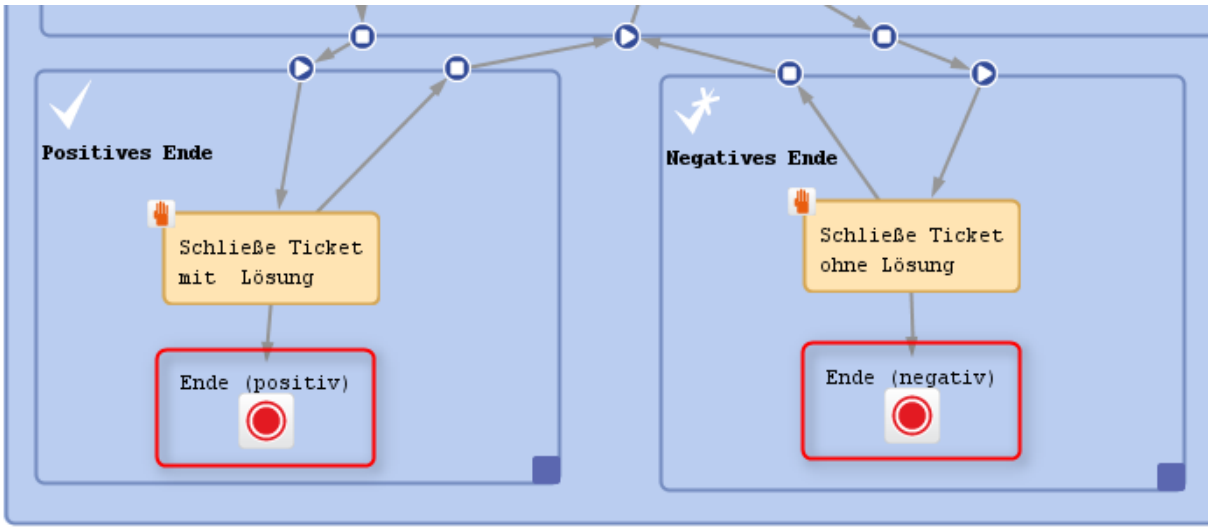


Abbildung 30: *ConSol CM Process Designer - Endknoten*

Ein Endknoten bildet das Schließen des Tickets ab, d. h. wenn ein Ticket in einen Endknoten wandert, ist es in technischem Sinne geschlossen.

C.3.1 Folgenden Aktionen sind für ein geschlossenes Ticket noch möglich

- Das Ticket kann von einem Administrator in der *Ticketverwaltung* des Admin Tools wieder geöffnet werden. Detaillierte Informationen dazu finden Sie im *ConSol CM Administratorhandbuch*.
- Andere Aktionen, die mit Administratorberechtigungen durchgeführt werden, können ebenfalls ausgeführt werden, d. h. Workflow-Aktionen sind noch möglich. Dies gilt zum Beispiel, wenn ein Trigger an dem Bereich hängt, in dem sich der Endknoten befindet. Der Trigger läuft weiter und misst die Zeit, auch wenn sich das Ticket im Endknoten befindet und geschlossen ist. Daher kann eine Aktion, die aufgrund eines gefeuerten Ereignissen durchgeführt wird, immer noch gestartet werden. Sie können dies vermeiden, indem Sie entweder keine Zeit-Trigger im entsprechenden Bereich verwenden oder den Endknoten außerhalb des Endbereichs platzieren.
  Außerdem kann ein anderes Ticket eine verknüpfte Aktion mit einem geschlossenen Ticket durchführen und es können Administrator-Task-Skripte (aus dem Task Execution Framework) ausgeführt werden.

C.3.2 Folgenden Aktionen sind für ein geschlossenes Ticket nicht möglich

- Bearbeiter können das Ticket nicht mehr editieren.

Sofern sie die erforderlichen Zugangsberechtigungen haben, können Bearbeiter das Ticket aber noch lesen. Dies ist die Grundlage für die Verwendung aller ConSol CM-Tickets des Systems als Wissenspool.

Die Weitergabe des Tickets an den Endknoten kann eine manuelle oder eine automatische Aktion sein. In der obigen Abbildung sind die Endknoten automatische Knoten, d. h. das Ticket wandert in diese Knoten, wenn die vorhergehende Workflow-Aktivität durchgeführt wurde.

Ein Workflow muss mindestens einen Endknoten enthalten, da es eine Möglichkeit geben muss, das Ticket zu schließen.

Sie können auch mehr als einen Endknoten erstellen. Dies kann hilfreich sein, um Reports zu erzeugen, z. B. um zwischen positiven und negativen Abschlüssen zu unterscheiden.

Ein Endknoten kann ein Skript haben, d. h. es kann ein Skript ausgeführt werden, bevor das Ticket geschlossen wird.

Manchmal kann es erforderlich sein, ein Ticket aus der Bearbeiterperspektive auf *geschlossen*, *abgeschlossen* oder *fertig* zu setzen, d. h. ein Ticket auf ein *vorläufiges Ende* zu setzen. Nach einer Zeit soll das Ticket dann automatisch geschlossen werden, wenn der Kunde keine weiteren Fragen oder Hinweise hatte. Dies können Sie erreichen, indem Sie einen Zeit-Trigger an die Endaktivität setzen und das Ticket nach der definierten Zeit automatisch zum Endknoten wandern lassen (siehe folgende Abbildung).

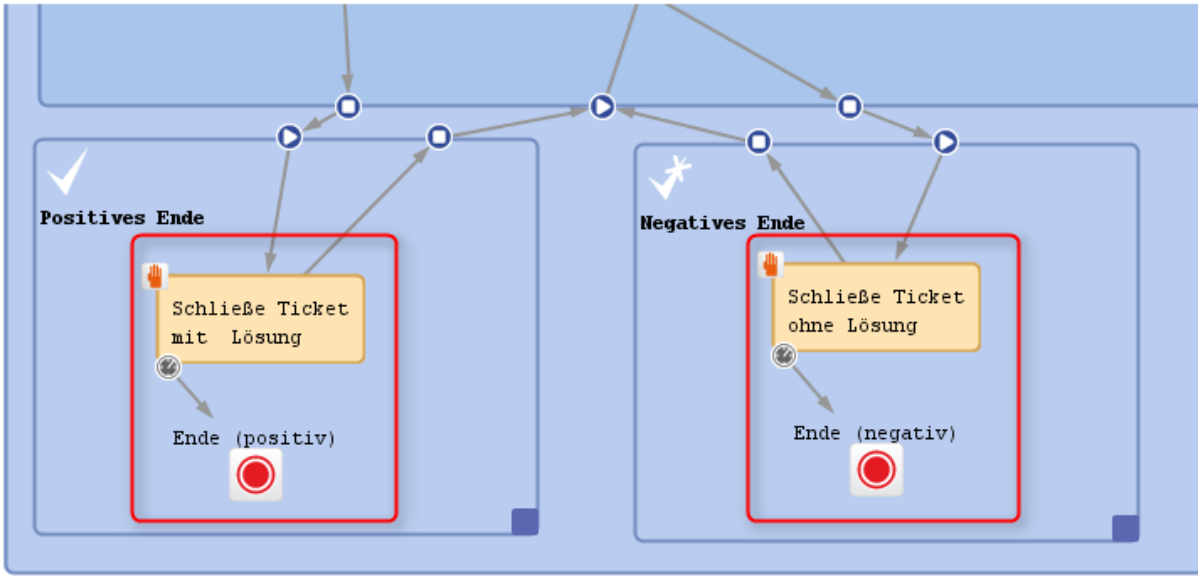


Abbildung 31: *ConSol CM Process Designer - Über Zeit-Trigger erreichte Endknoten*

C.3.3 Eigenschaften eines Endknotens

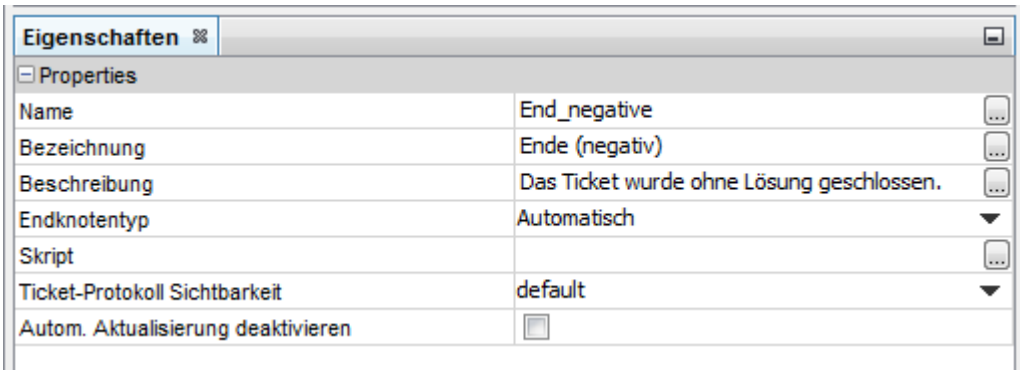


Abbildung 32: *ConSol CM Process Designer - Eigenschaften eines Endknotens*

Eigenschaften:

- **Name**
  Technischer Name des Objekts
- **Bezeichnung**
  Lokalisierter Name, der auf der GUI angezeigt wird.
- **Beschreibung**
  Beschreibung, die als Mouseover-Text angezeigt wird.
- **Endknotentyp**
  Automatisch/Manuell.
- **Skript**
  Hier kann ein Skript eingegeben werden, das ausgeführt werden soll, wenn das Ticket den Endknoten erreicht, d. h. bevor das Ticket geschlossen wird. Kann editiert werden.
- **Ticket-Protokoll Sichtbarkeit**
  Siehe Abschnitt Ticket-Protokoll Sichtbarkeit.
- **Autom. Aktualisierung deaktivieren**
  Siehe Abschnitt Autom. Aktualisierung deaktivieren.

C.4 Workflow-Komponenten: Bereiche (Scopes)

In diesem Kapitel werden folgende Themen behandelt:

C.4.1 Einführung in Bereiche

Wenn ein Ticket einen Prozess durchläuft, muss es mehrere Positionen in einer vordefinierten Reihenfolge durchlaufen. Zum Beispiel kann das Ticket in einer Servicedesk-Umgebung als *neues Ticket* hereinkommen und muss danach vorqualifiziert werden (in diesem Beispiel: Gibt es SLAs, die berücksichtigt werden müssen; handelt es sich um einen VIP-Kunden?). Danach kann der Bearbeiter an dem Ticket arbeiten und legt es möglicherweise für eine Zeit in die Wiedervorlage. Danach soll das Ticket geschlossen werden, und zwar entweder als *positiv, mit Lösung* oder *negativ, ohne Lösung*. Diese Hauptschritte des Prozesses sind in ConSol CM-Workflows als Bereiche abgebildet. Die folgende Abbildung zeigt einen Beispiel-Workflow.

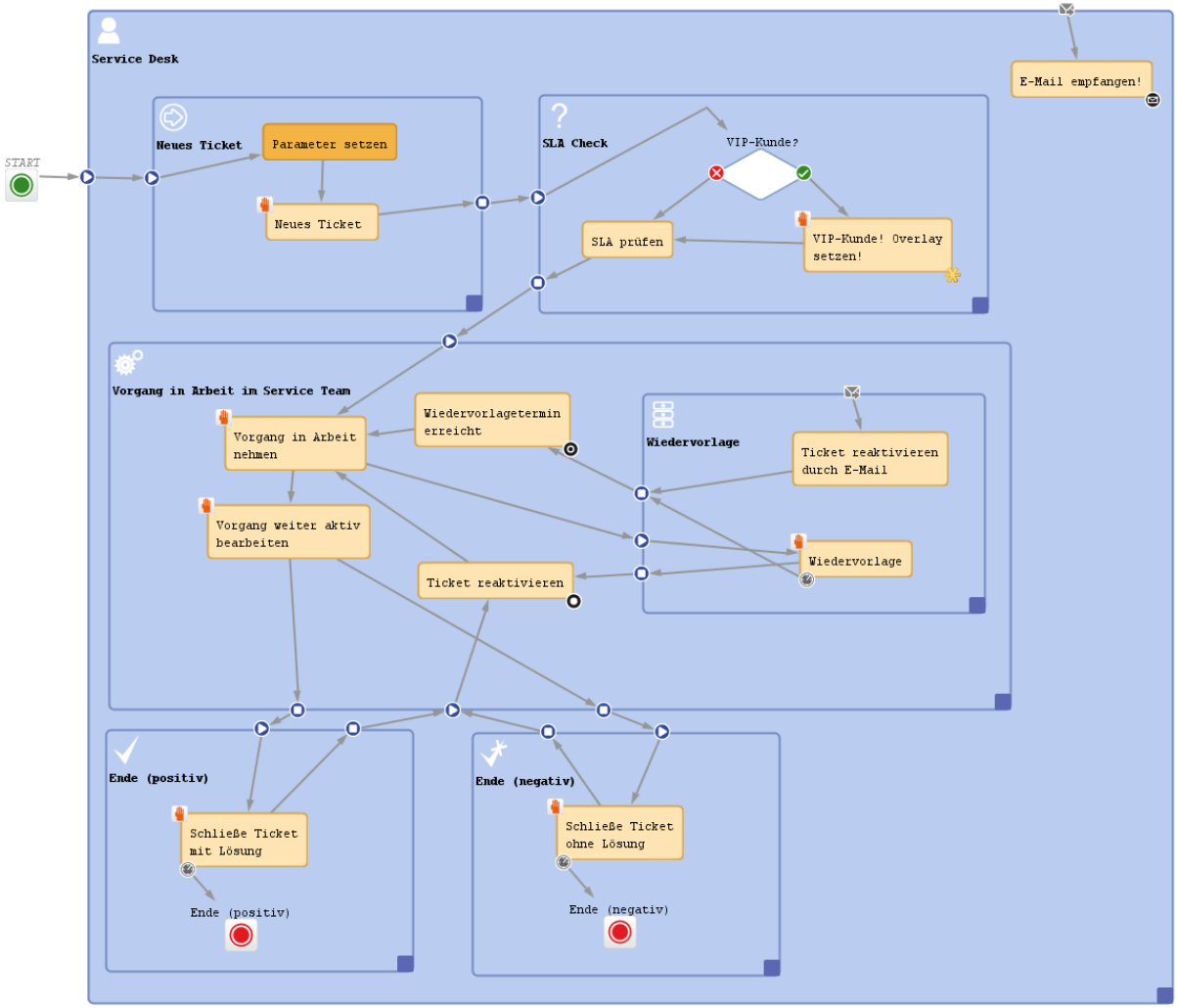


Abbildung 33: *ConSol CM Process Designer - Workflow mit Bereichen*

Innerhalb jedes größeren Prozessschritts (d. h. jedes Bereichs) kann es eine oder mehrere Aktivitäten geben, z. B. wird bei der Vorqualifizierung zuerst die VIP-Kundenprüfung und danach die SLA-Prüfung durchgeführt. Die Aktivitäten sind im Abschnitt Workflow-Komponenten: Aktivitäten detailliert beschrieben. An dieser Stelle werden nur die Bereiche erklärt.

Ein Bereich kann zu einem anderen Bereich gehören, oder - von der anderen Seite aus betrachtet - ein Bereich kann untergeordnete Bereiche enthalten.

Ein Bereich kann verschiedene Arten von Triggern haben, z. B. einen Mail-Trigger, der feuert, wenn ein Ticket, das sich gerade in diesem Bereich befindet, eine E-Mail erhält. Details dazu finden Sie in den Abschnitten Mail-Trigger, Zeit-Trigger und Event-Trigger.

C.4.2 Definieren eines neuen Bereichs

Um einen neuen Bereich zu definieren, d. h. einen neuen Bereich zum Workflow hinzuzufügen, ziehen Sie das Bereichs-Icon aus der Palette in den Workflow an die Stelle, an der Sie ihn positionieren möchten. Aktivieren Sie ihn mit einem Doppelklick. Danach können Sie neue Aktivitäten oder andere Elemente hinzufügen oder vorhandene Aktivitäten/Elemente in den Bereich ziehen. Wenn Sie die Elemente durch Zeichnen von Pfeilen verbinden, werden die Eintritts- und Austrittpunkte der Bereiche automatisch hinzugefügt.

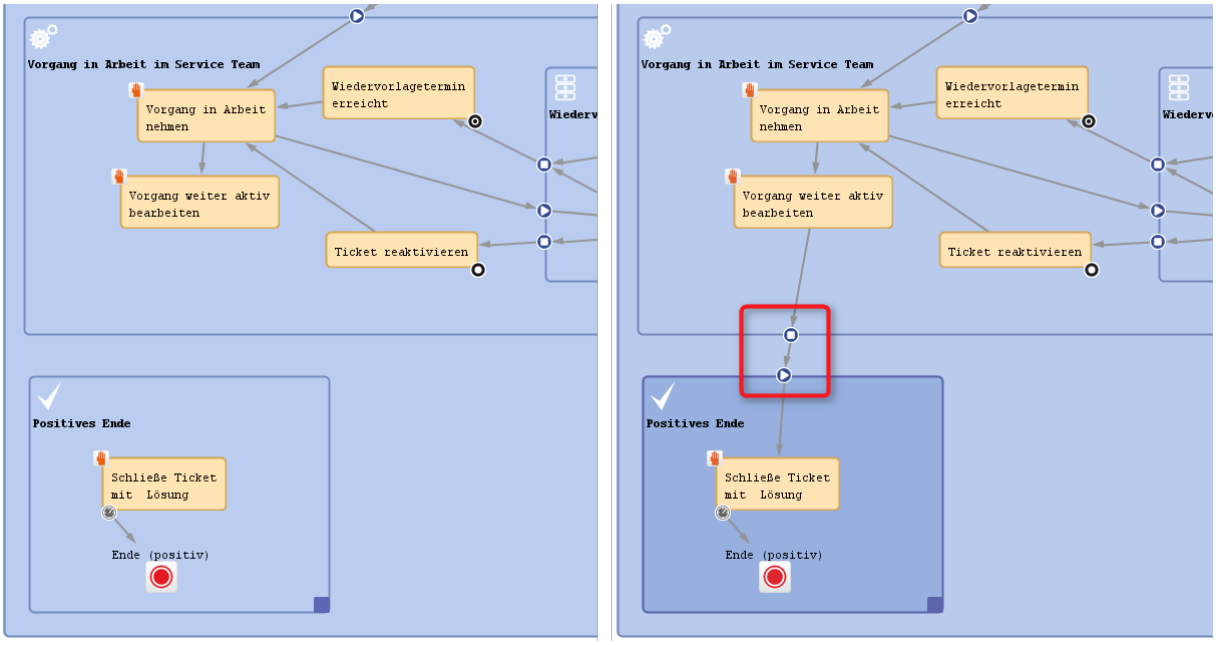


Abbildung 34: *ConSol CM Process Designer - Automatisch erzeugte Eintritts- und Austrittspunkte an Bereichen*

Wenn Sie den neuen Bereich hinzugefügt haben, können Sie die Eigenschaften des Bereichs definieren, siehe nächster Abschnitt.

C.4.3 Eigenschaften eines Bereichs

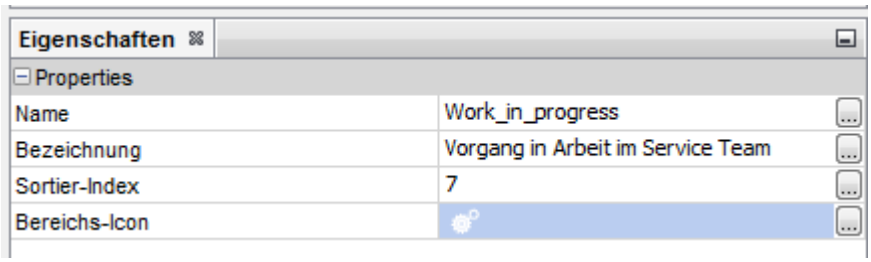


Abbildung 35: *ConSol CM Process Designer - Eigenschaften eines Bereichs*

Die folgenden Eigenschaften können für einen Bereich definiert werden:

- **Name**
  Technischer Name des Objekts
- **Bezeichnung**
  Lokalisierter Name, der auf der GUI des Web Clients angezeigt wird.
- **Sortier-Index**
  Definiert die Position der Tickets aus diesem Bereich in einer Sicht (sofern die Sicht mehr als einen Bereich umfasst) und beeinflusst die Reihenfolge in CM.Track V2, siehe Abschnitt Sortier-Index des Bereichs und seine Auswirkungen.
- **Bereichs-Icon**
  Das Icon, das als Bereichs-Icon auf der GUI des Web Clients angezeigt wird (siehe folgende Abbildung). Klicken Sie in das blaue Feld, um eines der Standard-Icons von ConSol CM auszuwählen, oder verwenden Sie den Dateibrowser (...), um ein Icon aus dem Dateisystem zu laden.

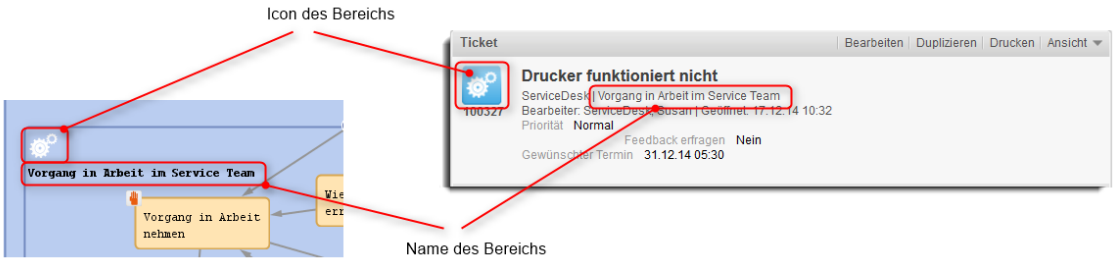


Abbildung 36: *ConSol CM Web Client - Bereichs-Icon*

> ⚠ Denken Sie daran, dass das Icon mit der Hintergrundfarbe des Ticket-Icons kombiniert wird. Deshalb sollten Sie (wenn Sie Ihre eigenen Icons hochladen möchten), transparente Bilder für die Ticket-Icons verwenden. Andernfalls geht die Hintergrundfarbe verloren oder ist nur in einem kleinen Rand um das Icon zu sehen.

C.4.4 Bereiche und Sichten

Sichten, d. h. die Auswahlkriterien für die Ticketliste, werden auf der Grundlage von Bereichen definiert. Eine detaillierte Erklärung von Sichten und der Sichtendefinition finden Sie im entsprechenden Abschnitt des *ConSol CM Administratorhandbuchs*.

Im aktuellen Kontext, d. h. bei der Definition von Bereichen im Workflow, ist es wichtig, im Hinterkopf zu behalten, welche Sichten später benötigt werden könnten. Der Mechanismus zur Unterscheidung von *neuen, aktiven* und *inaktiven* Tickets basiert zum Beispiel vollständig auf der Definition von Bereichen und Sichten:

- **Sicht: Neu**
  Alle neuen Tickets im Bereich *Neu*.
- **Sicht: Aktiv**
  Alle aktiven Tickets, d. h. Tickets, die sich **nicht** in Bereichen wie *Warten*, *Wiedervorlage* oder ähnlich befinden.
- **Sicht: Inaktiv**
  Alle Tickets, **die sich** in einem Bereich *Warten*, *Wiedervorlage* oder ähnlich befinden.

Dies bedeutet, dass, immer wenn eine Sicht benötigt wird, um eine bestimmte Art von Tickets anzuzeigen, dafür ein Bereich definiert werden muss.

⚠ Wir empfehlen Ihnen dringend, **keine** Sichten zu definieren, die geschlossene Tickets enthalten!

Die Anzahl der geschlossenen Tickets wird bei der Arbeit mit der Software stark zunehmen. Deswegen würden Sichten mit geschlossenen Tickets immer die (per System-Property definierte) maximal zulässige Anzahl an Tickets pro Sicht erreichen. Das kann negative Auswirkungen auf die Leistung des Web Clients haben und in den meisten Fällen werden die gesuchten Tickets noch nicht einmal zu den ersten 50 oder 100 Tickets gehören.

Schlussfolgerung: Eine Sicht mit geschlossenen Tickets hilft den Bearbeitern nicht weiter und kann die Systemgeschwindigkeit senken. Eine Sicht mit geschlossenen Tickets kann daher höchstens für Testumgebungen sinnvoll sein.

C.4.5 Sortier-Index des Bereichs und seine Auswirkungen

Der Sortier-Index definiert Folgendes:

- Die Sortierreihenfolge der Bereiche in CM.Track V2 (verfügbar in CM-Versionen 6.10.5 und höher)

C.5 Workflow-Komponenten: Aktivitäten

In diesem Kapitel werden folgende Themen behandelt:

C.5.1 Einführung in Aktivitäten

Eine Aktivität stellt eine Aktion im Workflow dar. Eine Aktivität befindet sich in einem Bereich und hat einen der folgenden Typen:

- manuell
- automatisch

Eine **manuelle** Aktivität muss durch eine manuelle Aktion des Bearbeiters im Web Client erfolgen. Die Aktivität wird im Web Client als *Workflow-Aktivität* angezeigt (sofern mindestens eine der Rollen des Bearbeiters die Berechtigung *Ausführen* hat (siehe *ConSol CM Administratorhandbuch*, Abschnitt *Rollenverwaltung*). Im Process Designer ist die Aktivität mit dem Icon *Hand* gekennzeichnet.

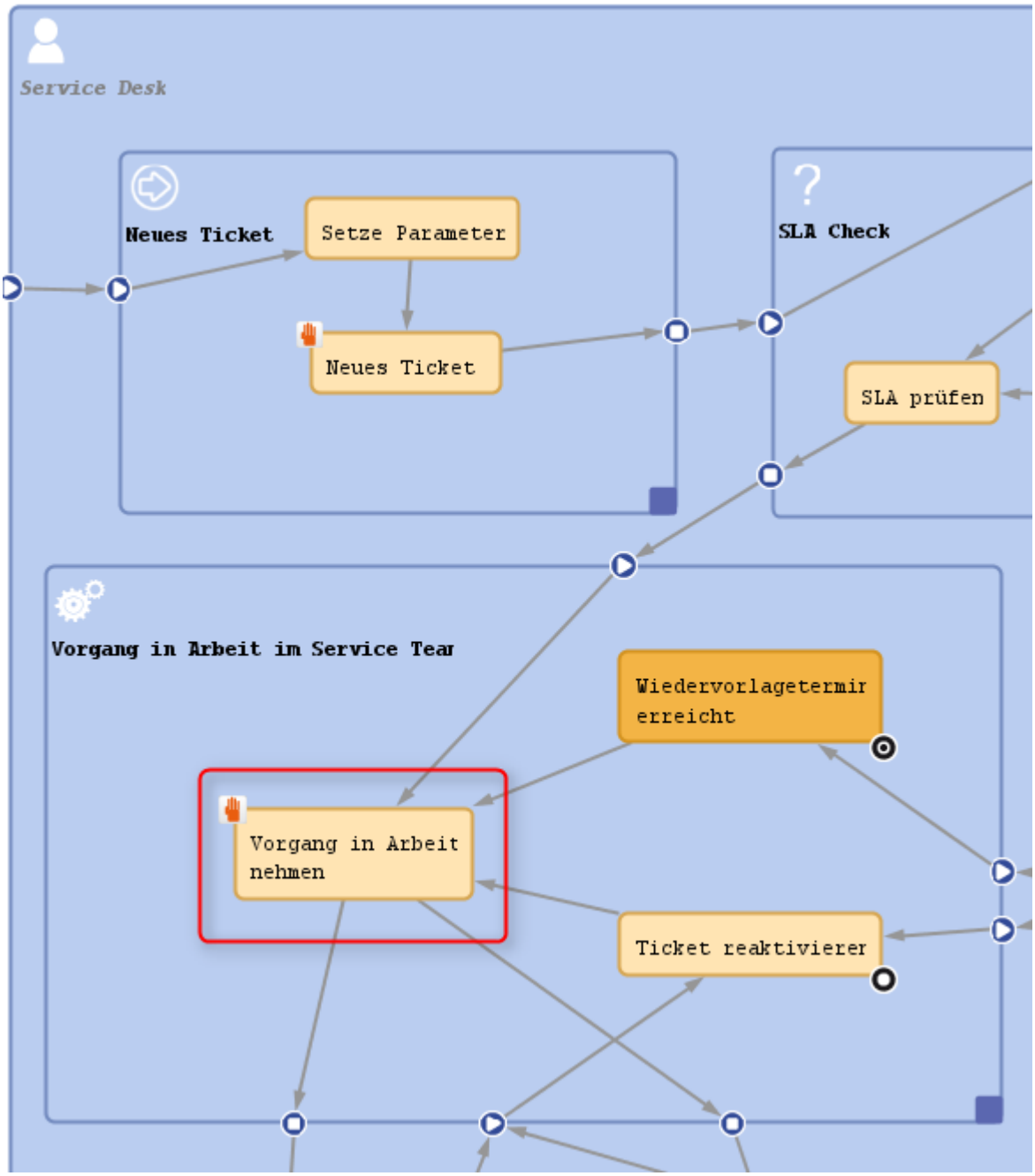


Abbildung 37: *ConSol CM Process Designer - Manuelle Aktivität im Workflow*

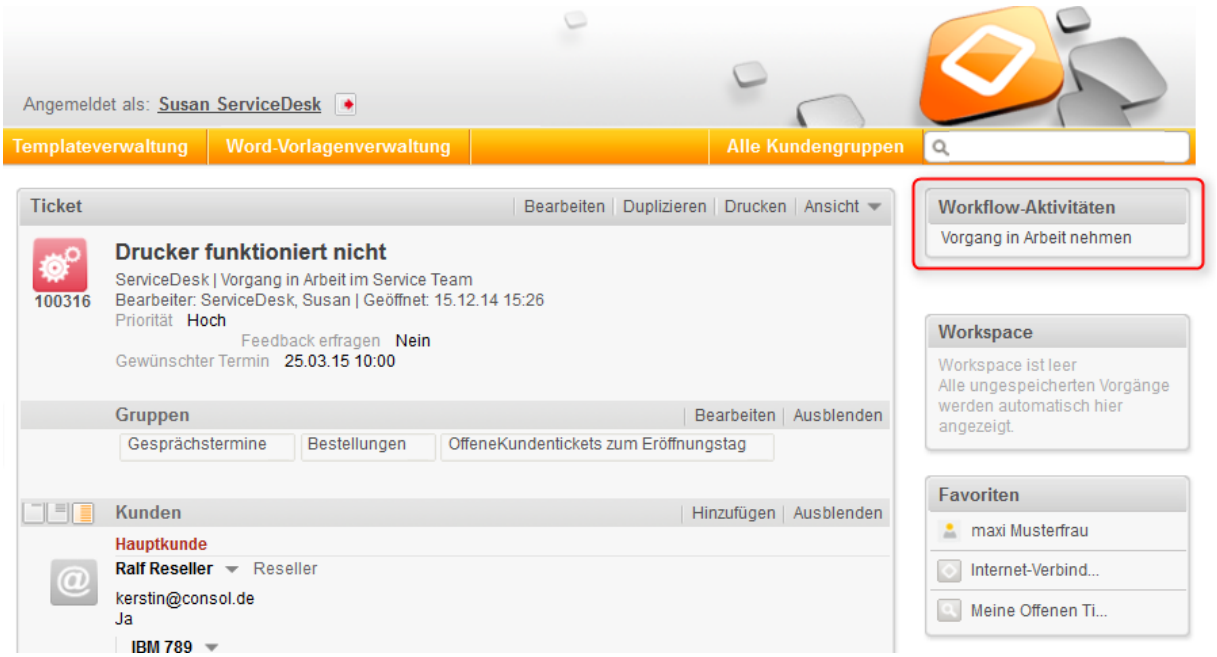


Abbildung 38: *ConSol CM Web Client - Manuelle Aktivität*

Eine **automatische** Aktivität wird automatisch vom System durchgeführt und nicht im Web Client angezeigt. Im Process Designer ist eine automatische Aktivität nicht mit einem speziellen Icon gekennzeichnet.

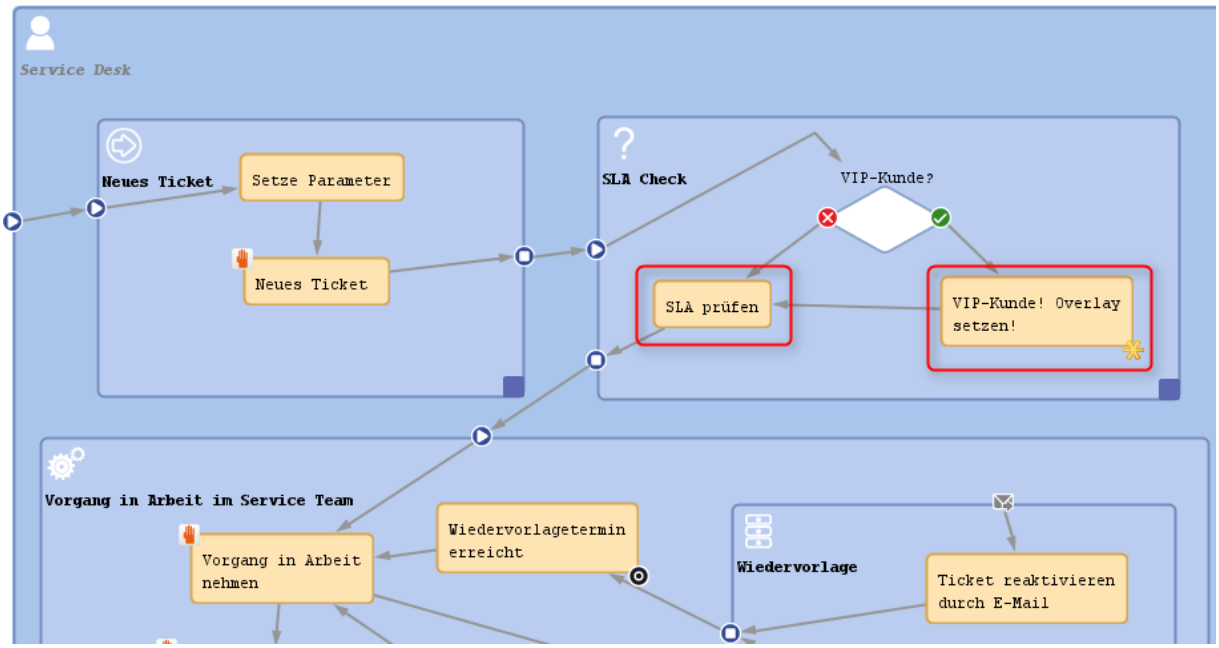


Abbildung 39: *ConSol CM Process Designer - Automatische Aktivitäten*

C.5.2 Eigenschaften einer Aktivität

Um die Eigenschaften einer Aktivität anzuzeigen und zu editieren, markieren Sie die Aktivität im Process Designer.

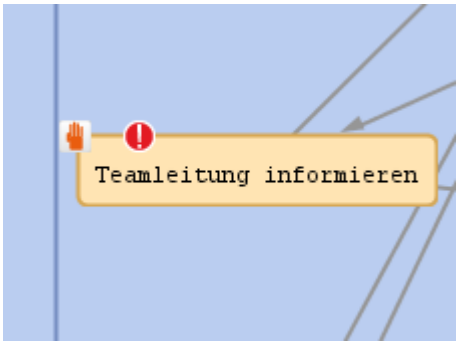


Abbildung 40: *ConSol CM Process Designer - Aktivität*

Der Eigenschaften-Editor wird für diese Aktivität geöffnet.

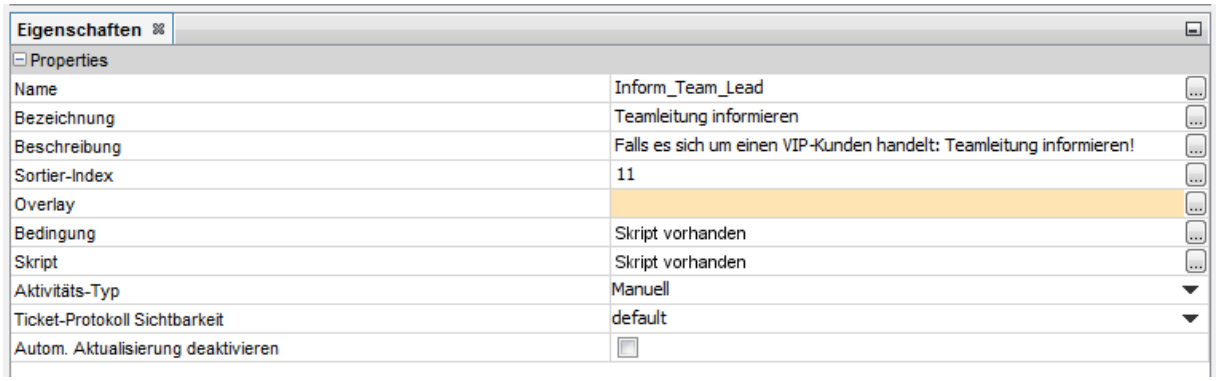


Abbildung 41: *ConSol CM Process Designer - Eigenschaften einer Aktivität*

Eine Aktivität kann folgende Eigenschaften haben:

- **Name**
  Pflichtangabe, technischer Objektname.
- **Bezeichnung**
  Optional (wenn nicht gesetzt, wird der technische Name verwendet). Lokalisierter Name, der im Web Client angezeigt wird. Dazu wird die Sprache verwendet, die im Webbrowser eingestellt ist.
- **Beschreibung**
  Optional. Wird im Web Client als Mouseover angezeigt.
- **Sortier-Index**
  Definiert die Reihenfolge der Aktivitäten im Web Client.

- **Overlay**
  Optional. Klicken Sie in das orangene Feld, um eines der Standard-Overlay-Icons von ConSol CM auszuwählen, oder verwenden Sie den Dateibrowser (...), um ein Icon aus dem Dateisystem hochzuladen.
- **Overlay-Gültigkeit**
  Wird nur angezeigt, wenn ein Overlay ausgewählt wurde:
  - **Aktivität**
    Das Overlay hängt so lange am Ticket-Icon, wie das Ticket hinter der Aktivität steht. Sobald die nächste Aktivität ausgeführt wird, wird das Overlay vom Ticket-Icon entfernt.
  - **Bereich**
    Das Overlay wird gelöscht, wenn das Ticket den Bereich verlässt.
  - **Prozess**
    Das einmal zum Ticket-Icon hinzugefügte Overlay hängt für den Rest des Prozesses am Ticket-Icon.
  - **Nächstes Overlay**
    Das Overlay hängt so lange am Ticket-Icon, wie kein neues Overlay hinzugefügt wird. Wenn ein neues Overlay hinzugefügt wird, wird das alte Overlay gelöscht.
- **Bedingung**
  Optional. Es kann ein Skript eingegeben werden, das ausgeführt wird, wenn die Aktivität im Web Client angezeigt werden soll. Das Skript muss *true* oder *false* zurückgeben. Wenn eine Bedingung für eine Aktivität definiert wurde, ist die Aktivität mit dem Icon *Ausrufezeichen* gekennzeichnet (siehe obige Abbildung).
  - Der Rückgabewert ist **true**.
    Die Aktivität wird angezeigt. Wenn es sich um eine manuelle Aktivität handelt, kann diese im Web Client vom Bearbeiter ausgewählt/durchgeführt werden.
  - Der Rückgabewert ist **false**.
    Die Aktivität wird nicht im Web Client angezeigt.

  > ⚠ CM-Version 6.9 und höher:
  >
  > Wenn Sie mit Datenobjektgruppenfeldern arbeiten, d. h. mit Datenfeldern, die Kundendaten enthalten, sollten Sie daran denken, dass möglicherweise die Datenmodelle von unterschiedlichen Kundengruppen berücksichtigt werden müssen, wenn ein Workflow für Queues verwendet wird, denen mehr als eine Kundengruppe zugewiesen ist!

- **Skript**
  Optional. Es kann ein Skript definiert werden, das ausgeführt wird, wenn das Ticket eine Aktivität durchläuft.

- **Aktivitäts-Typ**
  Pflichtangabe. Es muss entweder *automatisch* oder *manuell* ausgewählt werden. Wenn es sich um eine manuelle Aktivität handelt, ist die Aktivität auf der GUI des Process Designers mit dem Icon *Hand* gekennzeichnet.
- **Ticket-Protokoll Sichtbarkeit**
  Siehe Abschnitt Ticket-Protokoll Sichtbarkeit.
- **Autom. Aktualisierung deaktivieren**
  Siehe Abschnitt Autom. Aktualisierung deaktivieren.

C.5.3 Prozesslogik von Aktivitäten

Folgende Prozesslogik gilt für Aktivitäten:

1. Wenn ein Ticket eine Aktivität durchlaufen hat, wartet es nach dieser Aktivität (nicht vor der nächsten Aktivität!).
2. Wenn ein Ticket eine Aktivität durchlaufen hat, überprüft es, ob eine der möglichen Folgeaktivitäten eine automatische Aktivität ist. Wenn ja, durchläuft es auch diese automatische Aktivität. Deshalb kann es nur eine automatische Aktivität in einem Prozessschritt geben.
3. Das Ticket läuft automatisch durch (automatische) Aktivitäten, solange es neue automatische Aktivitäten gibt. Es hält an, sobald es eine oder mehrere manuelle Aktivitäten gibt, für die eine Aktion des Bearbeiters erforderlich ist.
4. Wenn eine oder mehrere der folgenden manuellen Aktivitäten ein Bedingungsskript haben, wird dieses Skript ausgeführt, um zu entscheiden, ob die Aktivität im Web Client angezeigt wird oder nicht.
5. Wenn der Bearbeiter die Aktivität im Web Client auswählt, wird das Skript der Aktivität ausgeführt.
6. Wenn es ein *postActivityScript* gibt, wird das Skript sofort nach der Ausführung des Aktivitätsskripts ausgeführt.
7. Das Ticket wartet nach der manuellen Aktivität. Wenn sich die folgende Aktivität in einem neuen Bereich befindet, tritt das Ticket nicht in den neuen Bereich ein. Es wartet immer nach der alten Aktivität und nicht vor der neuen Aktivität!

Sofern die Aktivität ein ACF hat, muss auch die Geschäftslogik von ACFs berücksichtigt werden.

> ⚠ Ein Ticket wartet immer nach der letzten Aktivität, die ausgeführt wurde, und nicht vor der nächsten Aktivität!

C.5.4 Beispiele für Aktivitäten

C.5.4.1 Beispiel 1: Bedingung für die Anzeige der Aktivität "Teamleitung informieren"

Falls ein Ticket von einem *VIP-Kontakt* geöffnet wurde, d. h. von einem Kontakt, bei dem das Boolean-Feld *vip* auf *true* gesetzt ist, soll die Teamleitung informiert werden. Wenn er kein *VIP* ist, soll die Aktivität nicht angeboten werden. Dafür wird das Datenobjektgruppenfeld *vip*, das zum Kundendatenmodell gehört, geprüft.

Abbildung 42: *ConSol CM Process Designer - Workflow-Aktivitäten (eine mit Bedingungsskript)*

```
// Get the main contact of the ticket. The unit object (can be a customer or a
 company) is provided;
// here it has to be a customer, i.e. a contact:

Unit contact = ticket.mainContact

// Check the Custom Field "vip" of the main contact. (see next image)
// If it is set to true, return true, i.e. the condition is TRUE.
// Else return false, i.e. the condition is FALSE:

if (contact.get("vip")) {
   return true
} else {
   return false
}
```

Code-Beispiel 2: *Bedingungsskript: Aktivität soll nur für VIP-Kunden angezeigt werden*

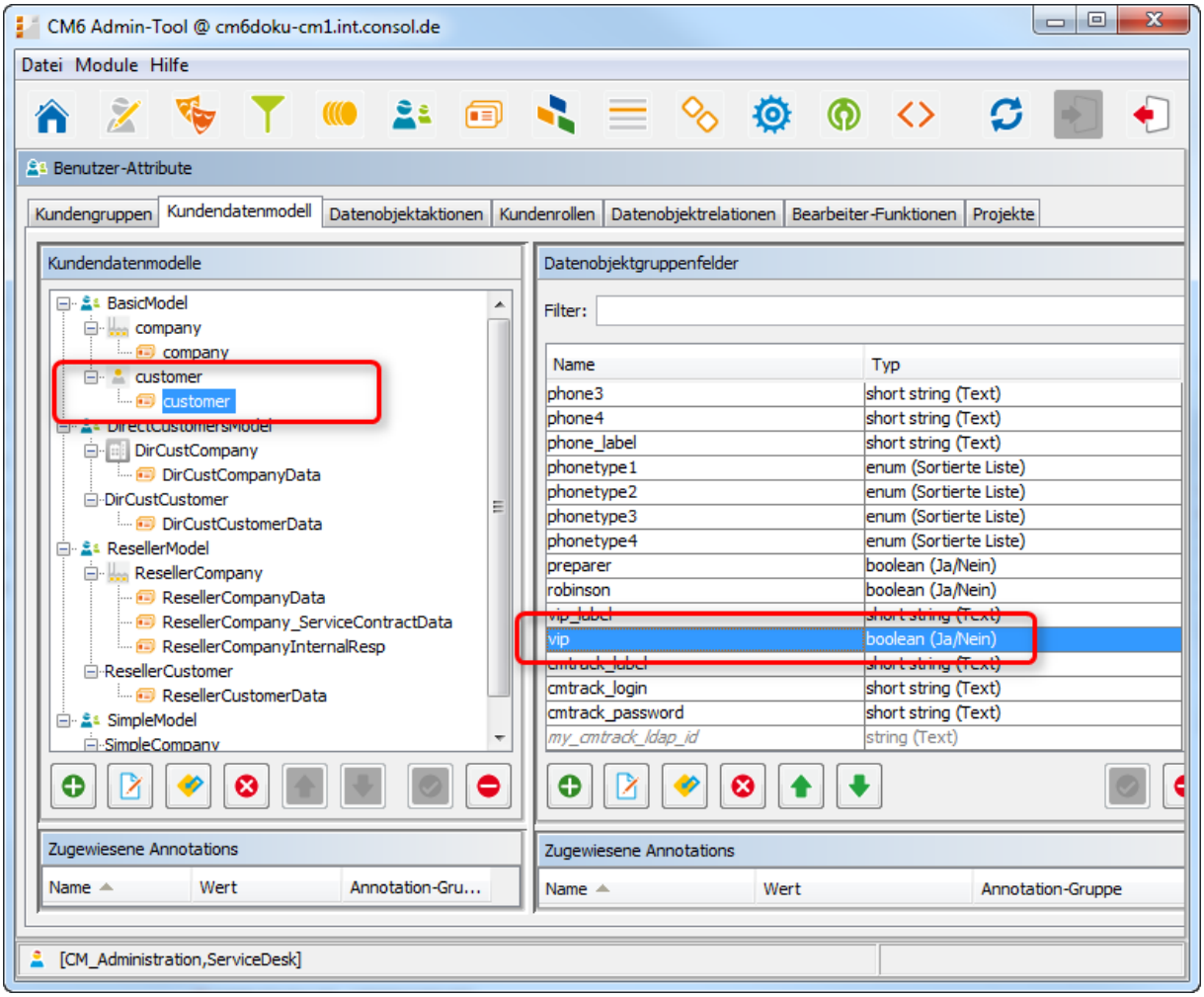


Abbildung 43: *ConSol CM Admin Tool - Datenobjektgruppenfeld "vip" (CM-Version 6.9)*

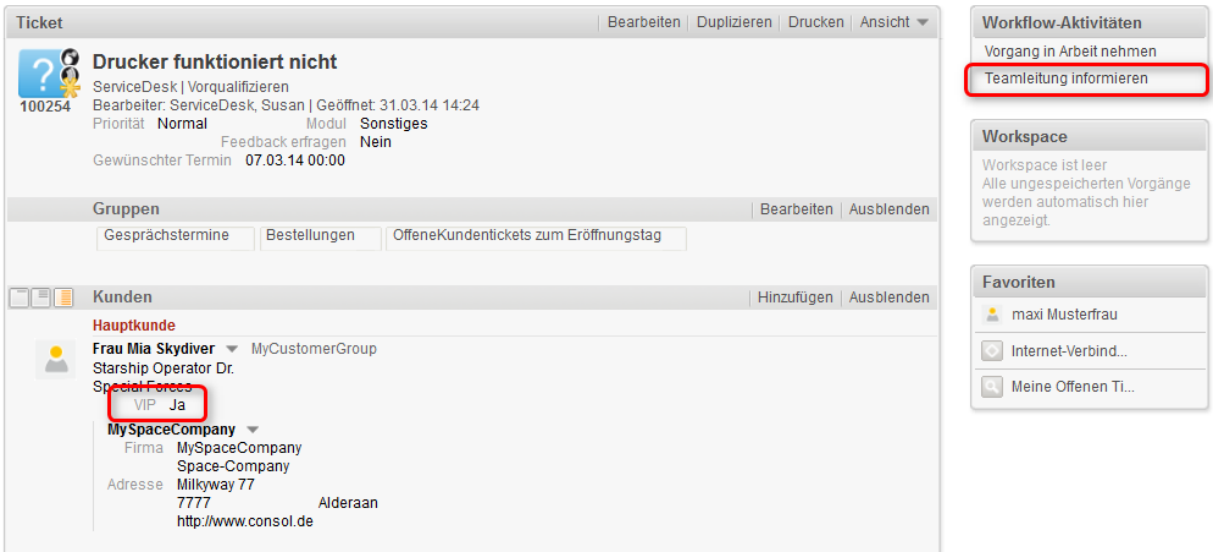


Abbildung 44: *ConSol CM Web Client - Bedingung: Rückgabewert TRUE*

Abbildung 45: *ConSol CM Web Client - Bedingung: Rückgabewert FALSE*

C.5.4.2 Beispiel 2: Senden einer E-Mail an den Hauptkontakt, wenn das Ticket erstellt wurde

Wenn ein Ticket erstellt wurde, soll eine E-Mail an den Hauptkontakt des Tickets gesendet werden.

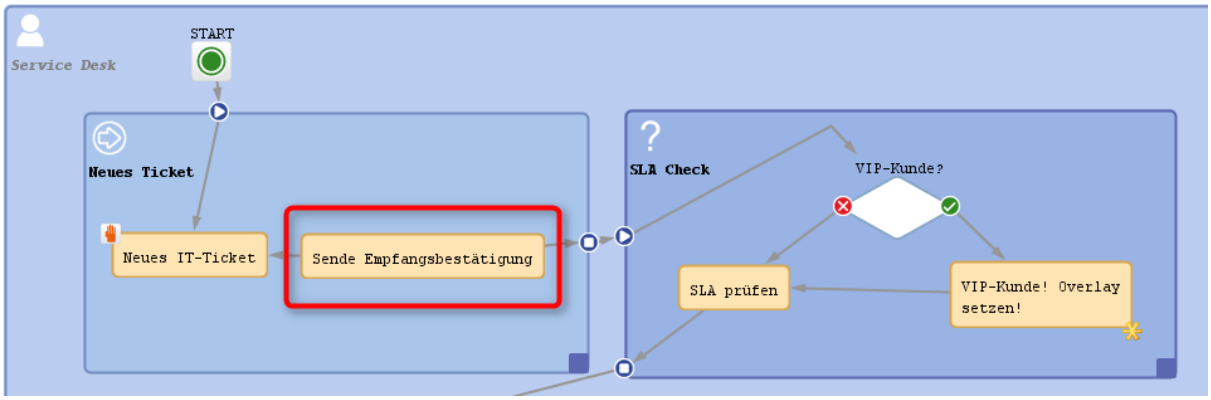


Abbildung 46: *ConSol CM Process Designer - Automatische Aktivität, bei der eine Empfangsbestätigung gesendet wird*

```
// Get the main contact of the ticket:
def mycontact = ticket.mainContact

// Get the value of the Custom Field "email" of the main contact:
def mycontact_e = mycontact.get("email")

// Use as text the e-mail template with name "receipt_notice_ServiceDesk".
// Can be located in the Template Designer or in the Admin-Toool.
// Usually e-mail templates are stored in the Template Designer:
def text = workflowApi.renderTemplate("receipt_notice_ServiceDesk")

// Get the reply-to address for the e-mail.
// This is stored in the system property "cmweb-server-adapter","mail.reply.to":
def replyto = configurationService.getValue("cmweb-server-adapter","mail.reply.to")


// Build the string for the ticket subject.
// Keep in mind that the regular expression which defines the ticket identifier has
 to be in this subject.
// Otherwise, an e-mail cannot be assigned to the correct ticket.
def subj = "Your request has been received: ticket (" + ticket.getId() + ")"

//Send out the e-mail
workflowApi.sendEmail(contact_e,subj,text,replyto,null)
```

Code-Beispiel 3: *Skript für automatische Aktivität, bei der eine Empfangsbestätigung gesendet wird, Variante 1*

```
// all lines of code identical to variant 1 except for the last line:

new Mail().setSubject( subj ).setTo( contact_e ).setReplyTo( replyto ).setText(
 text ).setTicketAttachments( null ).send()
```

Code-Beispiel 4: *Skript für automatische Aktivität, bei der eine Empfangsbestätigung gesendet wird, Variante 2*

C.5.4.3 Beispiel 3: Zuweisen des Tickets an den aktuellen Bearbeiter

Das Ticket soll dem Bearbeiter zugewiesen werden, der die Aktivität *Neues IT-Ticket* durchführt.

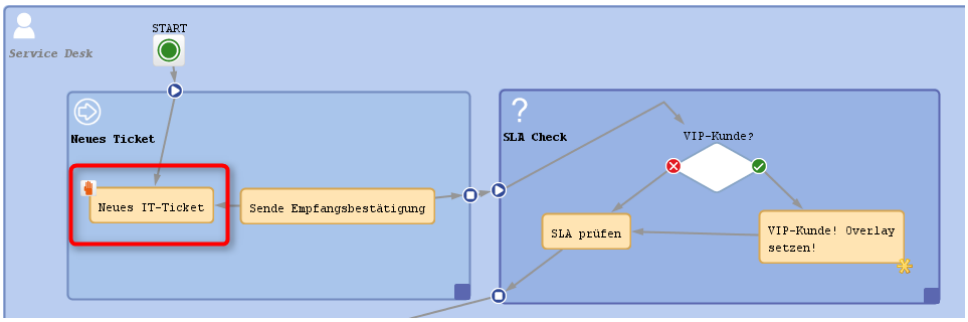


Abbildung 47: *ConSol CM Process Designer - Workflow-Aktivität, bei der der Bearbeiter zugewiesen werden soll*

Abbildung 48: *ConSol CM Web Client - Ticket hat die Aktivität, bei der der Bearbeiter zugewiesen wird, durchlaufen*

```
// Get the engineer who is executing the activity:
def curr_eng = workflowApi.getCurrentEngineer()

// Assign the ticket to the current engineer
ticket.setEngineer(curr_eng)
```

Code-Beispiel 5: *Skript zum Zuweisen des Tickets an den aktuellen Bearbeiter*

Stellen Sie sicher, dass Sie immer das richtige Bearbeiterobjekt verwenden!

Der aktuelle Bearbeiter ist der Bearbeiter, der angemeldet ist und die aktuelle Aktivität durchführt. Sie können das Objekt mit einer der folgenden Methoden abrufen.

Mit workflowApi:

```
// Java notation
def curr_eng = workflowApi.getCurrentEngineer()
```

```
// Groovy notation
def curr_eng = workflowApi.currentEngineer
```

Mit engineerService:

```
// Java notation:
def curr_eng = engineerService.getCurrent()
```

```
//Groovy notation
def curr_eng = engineerService.current
```

Der Ticketbearbeiter ist die Person, der das Ticket (zu diesem Zeitpunkt) zugewiesen ist und die für das Ticket verantwortlich ist. Sie können das Objekt mit der folgenden Methode abrufen:

```
//Java notation
def tic_eng = ticket.getEngineer()
```

```
//Groovy notation
def tic_eng = ticket.engineer
```

C.6 Workflow-Komponenten: Entscheidungsknoten

In diesem Kapitel werden folgende Themen behandelt:

C.6.1 Einführung in Entscheidungsknoten

Ein Entscheidungsknoten ist ein Knoten, der einen oder mehrere Eintrittspunkte und genau zwei Austrittspunkte hat: *true* und *false*. Ein Entscheidungsknoten muss immer ein Skript haben, das entweder *true* oder *false* zurückgibt.

Das Ticket wandert in den Entscheidungsknoten, dann wird das Skript ausgeführt und abhängig vom Ergebnis (*true* oder *false*) verlässt das Ticket den Entscheidungsknoten über den entsprechenden Austrittspunkt.

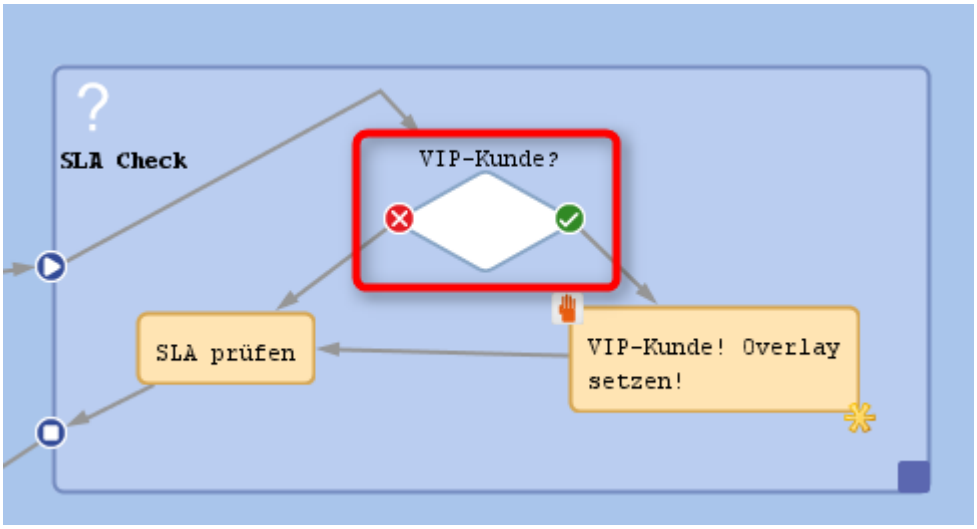


Abbildung 49: *ConSol CM Process Designer - Entscheidungsknoten*

C.6.2 Eigenschaften eines Entscheidungsknotens

Ein Entscheidungsknoten hat folgende Eigenschaften:

- **Name**
  Pflichtangabe, technischer Objektname.
- **Bezeichnung**
  Optional, der lokalisierte Name, der auf der GUI des Web Clients angezeigt wird.
- **Bedingung**
  Pflichtangabe, ein Skript, das *true* oder *false* zurückgibt, muss angegeben werden.
- **Ticket-Protokoll Sichtbarkeit**
  Siehe Abschnitt Ticket-Protokoll Sichtbarkeit.
- **Autom. Aktualisierung deaktivieren**
  Siehe Abschnitt Autom. Aktualisierung deaktivieren.

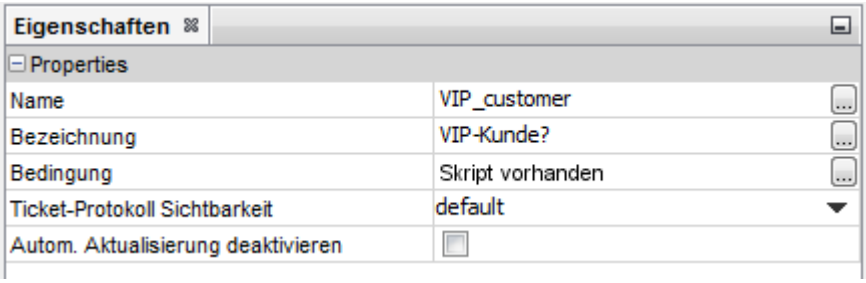


Abbildung 50: *ConSol CM Process Designer - Entscheidungsknoten: Eigenschaften*

C.6.3 Beispiel für einen Entscheidungsknoten

Im folgenden Beispiel soll das System automatisch überprüfen, ob der Kunde (Hauptkunde des Tickets) ein *VIP-Kunde* ist. Wenn ja, soll das Ticket das Overlay *VIP* erhalten (im Beispiel einen gelben Stern).

1. Ein Datenobjektgruppenfeld des Typs *boolean (ja/nein)* muss im Kundendatenmodell (*FlexCDM*) definiert werden, um den Kunden als *VIP* (ja/nein) zu markieren. Siehe auch *ConSol CM Administratorhandbuch*, Abschnitt *Einrichten des Kundendatenmodells*.

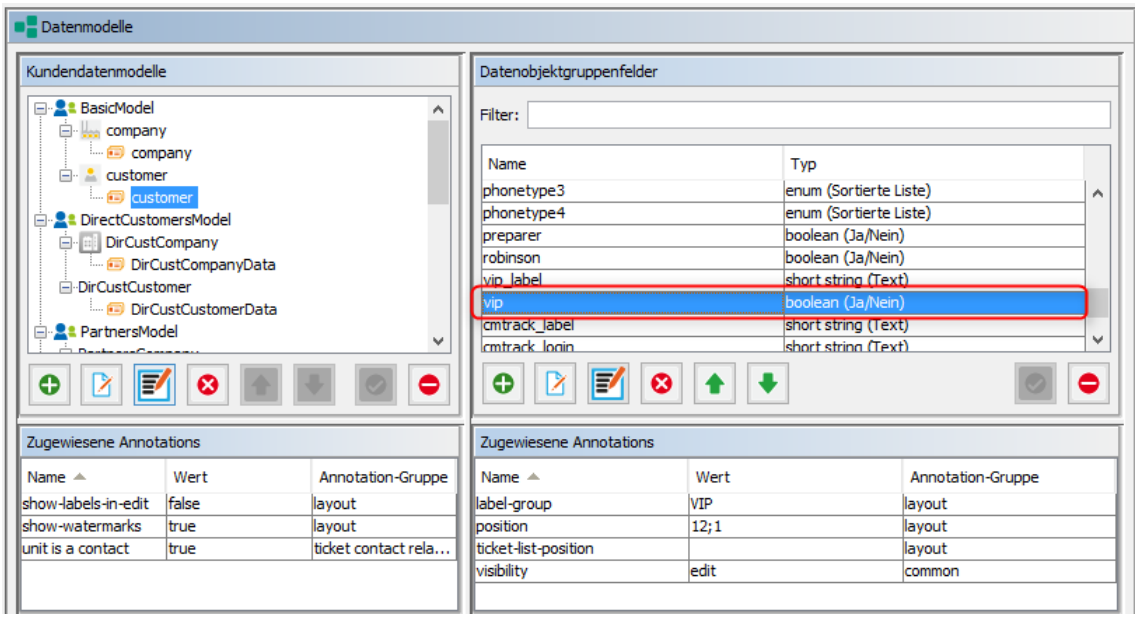


Abbildung 51: *ConSol CM Admin Tool - Datenobjektgruppenfeld "vip" in den Kundendaten*

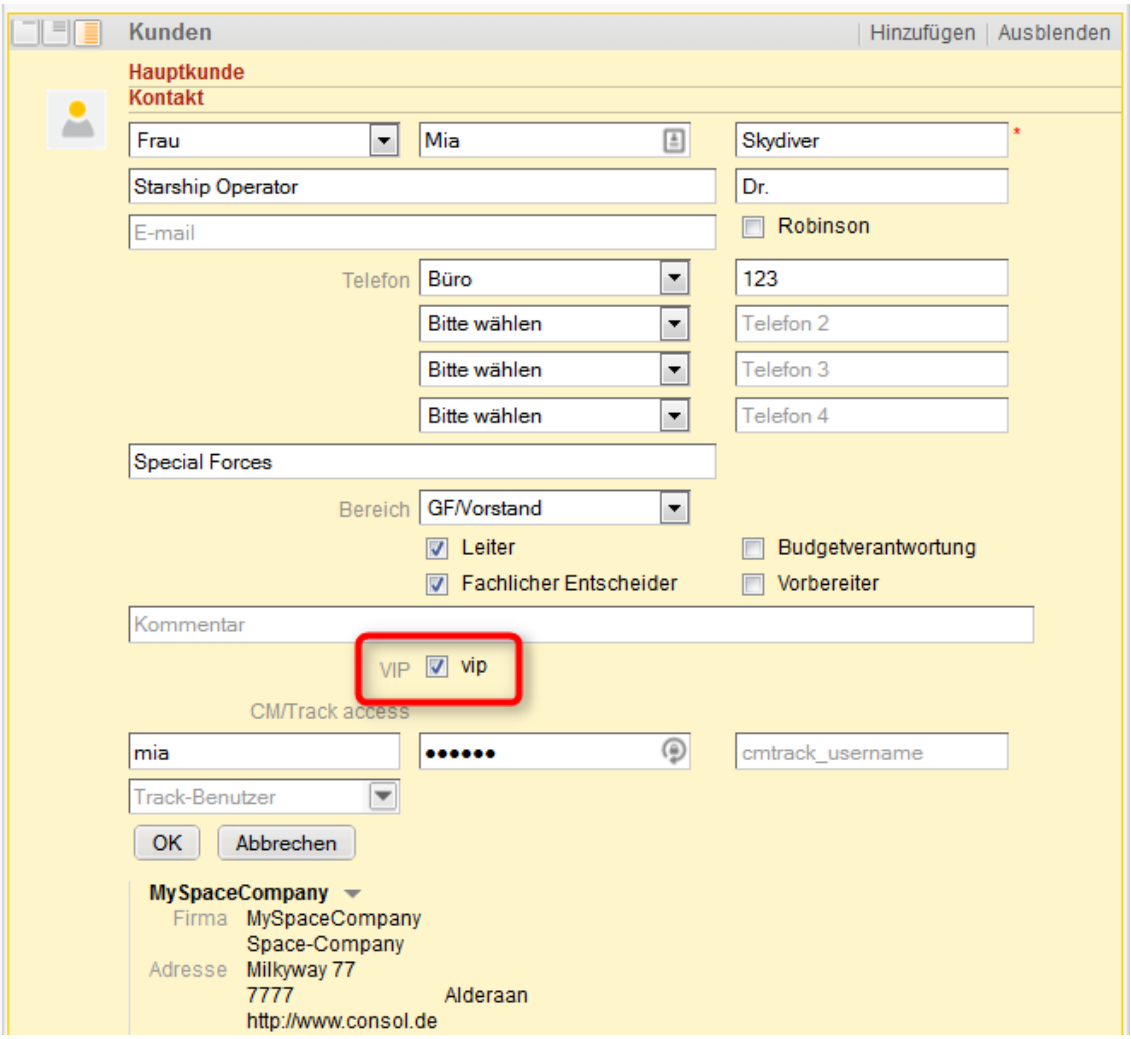


Abbildung 52: *ConSol CM Web Client - Datenobjektgruppenfeld "VIP" für Kunden-/Kontaktdaten*

2. Im Skript des Entscheidungsknotens muss überprüft werden, ob der Kunde ein *VIP* (Rückgabewert: *true*) ist oder nicht (Rückgabewert: *false*).

```
// Get the main contact of the ticket. The unit object (can be a customer or
 a company) is provided;
// here it has to be a customer, i.e. a contact:

Unit mycontact = ticket.mainContact

// Check the Custom Field "vip" of the main contact. (see next image)
// If it is set to true, return true, i.e. the condition is TRUE.
// Else return false, i.e. the condition is FALSE:

if (mycontact.get("vip")) {
   return true
} else {
   return false
}
```

Code-Beispiel 6: *Arbeiten mit Kundendaten. Beispiel: Verwendung eines Datenobjektgruppenfeldes des Typs boolean*

3. Wenn ein Ticket den Entscheidungsknoten und die darauf folgende automatische Aktivität durchlaufen hat, bei der das Overlay *VIP* hinzugefügt wurde, erhält das Ticket-Icon im Web Client das Overlay, siehe folgende Abbildung.

Abbildung 53: *ConSol CM Web Client - Ticket-Icon mit VIP-Overlay*

C.7 Adornments (Trigger und ACFs)

Die ConSol CM-Workflow-Engine kann auf unterschiedliche Arten von Ereignissen reagieren. Dies wird durch Trigger gesteuert. ACFs sind dynamische Formulare.

Adornment-Typ	Erklärung
Zeit-Trigger	Kontrolliert die Zeit, die vergangen ist, seit das Ticket in den Bereich oder die Aktivität gelangt ist, siehe Abschnitt Zeit-Trigger.
Mail-Trigger	Kontrolliert, ob ein Ticket, das sich in dem Bereich befindet, eine E-Mail Ticket empfangen hat, siehe Abschnitt Mail-Trigger.
Event-Trigger	Kontrolliert Ereignisse wie den Wechsel des Bearbeiters oder das Hinzufügen eines Kommentars. Siehe Abschnitt Event-Trigger.
ACF	Mit Aktivitäts-Formularen (ACFs) können Sie die Daten kontrollieren, die der Benutzer an einem bestimmten Schritt des Prozesses eingeben muss, siehe Abschnitt Aktivitäts-Formulare (ACFs).

C.7.1 Zeit-Trigger

In diesem Kapitel werden folgende Themen behandelt:

- Einführung in Zeit-Trigger
- Hinzufügen eines Zeit-Triggers zu einem Workflow
- Eigenschaften eines Zeit-Triggers
- Geschäftslogik und Initialisierung eines Zeit-Triggers
- Beispiele für Zeit-Trigger
- Skripte mit Zeit-Triggern

C.7.1.1 Einführung in Zeit-Trigger

Ein Workflow kann mehrere Zeit-Trigger enthalten.

Abbildung 54: *ConSol CM Process Designer - Zeit-Trigger*

Ein Zeit-Trigger ist ein Mechanismus, der reagiert, wenn ein bestimmter Zeitraum abgelaufen ist. Dies kann zum Beispiel in folgenden Situationen erforderlich sein:

- **Anwendungsfall 1:**
  Ein Bearbeiter möchte ein Ticket für eine bestimmte Zeit auf Wiedervorlage legen, da er weiß, dass der Kunde bis dahin nicht erreichbar ist.
- **Anwendungsfall 2:**
  Das System soll die Eskalationszeit automatisch kontrollieren, d. h. wenn ein Ticket eingeht und nicht bearbeitet wird, soll es eine Warnung geben (dies kann ein Overlay am Ticket-Icon, eine E-Mail an den Teamleiter oder eine andere Aktion sein).
- **Anwendungsfall 3:**
  Ein Ticket wurde gelöst und der Bearbeiter schließt es. Dies soll allerdings nur das vorläufige Ende sein und das Ticket soll nach einem bestimmten Zeitraum technisch geschlossen werden.

Diese Anwendungsfälle können mittels Zeit-Triggern implementiert werden.

Ein Zeit-Trigger kann so konfiguriert werden, dass er einen Arbeitszeitkalender verwendet, d. h. nur die Zeiten berücksichtigt, die als Arbeitszeiten definiert sind.

Ein Zeit-Trigger kann hängen an ...

- **einem Bereich**
  Dann kontrolliert er alle Tickets, die sich gerade in diesem Bereich befinden.
- **einer Aktivität**
  Dann kontrolliert er nur die Tickets, die diese Aktivität gerade durchlaufen haben.

Ein Zeit-Trigger muss einen der beiden folgenden Typen haben:

- manuell
- mit einer definierten Zeitspanne

Sie als Workflow-Entwickler müssen alles implementieren, was geschehen soll, wenn ein Zeit-Trigger gefeuert hat! Es gibt keine automatischen Aktionen. Der Zeit-Trigger sorgt lediglich für das Signal *Zeit abgelaufen*, genauso wie ein Wecker.

C.7.1.2 Hinzufügen eines Zeit-Triggers zu einem Workflow

Hinzufügen eines Zeit-Triggers zu einem Bereich

Klicken Sie in der Palette auf das Zeit-Trigger-Symbol und ziehen Sie es in den gewünschten Bereich. Es wird automatisch an den oberen Rand des Bereichs angefügt. Sie können die Position später ändern (ziehen Sie den Trigger nach links oder rechts, um die Reihenfolge der Trigger zu ändern oder das Layout zu verbessern).

Ein Zeit-Trigger, der an einem Bereich hängt, kann nicht in einen anderen Bereich oder eine Aktivität verschoben werden. Wenn Sie einen Zeit-Trigger an einen anderen Bereich oder eine Aktivität anfügen möchten, entfernen Sie den bereits erstellen Trigger und erstellen Sie einen neuen Trigger für den richtigen Bereich bzw. die Aktivität.

Um die Eigenschaften des Triggers zu konfigurieren, wählen Sie ihn im Bearbeitungsbereich aus und setzen Sie im Eigenschaften-Editor die richtigen Werte. Siehe Abschnitt Eigenschaften eines Zeit-Triggers.

Sie können vom Trigger ausgehend Verbindungen zu Aktivitäten oder Entscheidungsknoten ziehen, die sich hinter dem Trigger befinden sollen. Der erste Schritt, der nach dem Zeit-Trigger ausgeführt wird, muss immer eine automatische Aktivität sein!

Hinzufügen eines Zeit-Triggers zu einer Aktivität

Klicken Sie in der Palette auf das Zeit-Trigger-Symbol und ziehen Sie es in die gewünschte Aktivität. Es wird an die Ecke der Aktivität angefügt.

Ein Zeit-Trigger, der an einer Aktivität hängt, kann nicht in eine anderen Aktivität oder einen Bereich verschoben werden. Wenn Sie einen Zeit-Trigger an eine andere Aktivität oder einen Bereich anfügen möchten, entfernen Sie den bereits erstellen Trigger und erstellen Sie einen neuen Trigger für die richtige Aktivität bzw. den Bereich.

Um die Eigenschaften des Triggers zu konfigurieren, wählen Sie ihn im Bearbeitungsbereich aus und setzen Sie im Eigenschaften-Editor die richtigen Werte. Siehe Abschnitt Eigenschaften eines Zeit-Triggers.

Sie können vom Trigger ausgehend Verbindungen zu Aktivitäten oder Entscheidungsknoten ziehen, die sich hinter dem Trigger befinden sollen. Der erste Schritt, der nach dem Zeit-Trigger ausgeführt wird, muss immer eine automatische Aktivität sein!

C.7.1.3 Eigenschaften eines Zeit-Triggers

Ein Zeit-Trigger hat folgende Eigenschaften:

- **Name**
  Pflichtfeld. Der technische Name des Triggers. Er wird automatisch gesetzt, kann aber manuell geändert werden.
- **Minuten/Stunden/Tage**
  Hier können Sie den Zeitraum angeben, nach dem der Trigger feuern soll. Die Anzeige bezieht sich immer auf einen 24-Stunden-Tag, d. h. wenn Sie 30 Stunden als Reaktionszeit eingeben und den Workflow erneut öffnen, steht dort 1 Tag, 6 Stunden.
- **Kalender**
  Optional. Markieren Sie diese Checkbox, wenn der Arbeitszeitkalender bei der Berechnung des Zeitraums berücksichtigt werden soll.

  Denken Sie daran, dass drei Schritte notwendig sind, um sicherzustellen, dass Zeiträume mit einem Arbeitszeitkalender berechnet werden:

  1. Definieren Sie einen Arbeitszeitkalender (siehe *ConSol CM Administratorhandbuch*, Abschnitt *Arbeitszeitkalender*).
  2. Weisen Sie den richtigen Arbeitszeitkalender einer Queue zu (siehe *ConSol CM Administratorhandbuch*, Abschnitt *Queue-Verwaltung*).
  3. Markieren Sie die Checkbox *Kalender* für jeden Trigger, der den Kalender benutzen soll.

  Prinzip der Verwendung eines Arbeitszeitkalenders:

  1 Tag bedeutet 24 Stunden absolute Zeit; dies hat nichts mit der Verwendung eines Kalenders zu tun. Der Kalender spielt nur eine Rolle, wenn der Zeit-Trigger aktiviert ist. Dann werden die 24 Stunden, d. h. 86400000 Millisekunden, als Eingabe für den Arbeitszeitkalender verwendet (sofern der Kalender aktiviert ist).

  Beispiel:

  Wenn der Zeitraum 1 Tag = 24 Stunden ohne Kalender ist, werden die 24 Stunden wie reguläre Zeit berechnet. Die Eskalation wird also einen Tag später zur gleichen Uhrzeit ausgelöst.

  Im Gegensatz dazu: Wenn ein Kalender verwendet wird (mit zum Beispiel 7 Arbeitsstunden pro Arbeitstag), werden die 24 Stunden entsprechend dem Kalender aufgeteilt. Der Trigger feuert also über 3 Tage später (24 Stunden = 3 x 7 Stunden + 3 Stunden).

  Siehe auch Abschnitt Arbeiten mit Kalendern und Zeiten.
- **wiederholbar**
  Optional. Markieren Sie diese Checkbox, um sicherzustellen, dass der Trigger mehr als einmal für ein Ticket feuern kann. Wenn ein Trigger *wiederholbar* ist, wird er, sofort nachdem er gefeuert hat, zurückgesetzt, d. h. die Zeitmessung beginnt erneut.

> Das Skript zu Beginn wird erneut ausgeführt. Das erste Feuern des Triggers wird durch den (technischen) Benutzer *admin* ausgelöst, alle folgenden durch den *Job Executor*.

- **Skript nach Ablauf**
  Optional. Es kann ein Skript definiert werden, das ausgeführt wird, wenn das Zeitintervall, das durch den Trigger kontrolliert wird, abgelaufen ist, d. h. wenn der Trigger feuert.
- **Skript zu Beginn**
  Optional. Es kann ein Skript definiert werden, das ausgeführt wird, wenn der Zeit-Trigger anfängt, die Zeit zu messen, d. h. wenn das Ticket in den Bereich bzw. die Aktivität eingetreten ist, an dem/der der Trigger hängt.
- **manuell**
  Optional, nur für Zeit-Trigger an Aktivitäten. Markieren Sie diese Checkbox, wenn der Bearbeiter die Zeit auswählen soll, nach der der Trigger feuern soll. Für den Bearbeiter wird ein Datumsauswahl-Feld (Kalender) angezeigt.
- **Wiederholungsintervall**
  Die Zeit in Sekunden, nach der die Trigger-Ausführung erneut ausgeführt werden soll, für den Fall, dass ein Skript in einen Fehler gelaufen ist. Die Zeit kann im Admin Tool konfiguriert werden (Property *jobExecutor.timerRetryInterval.seconds*).

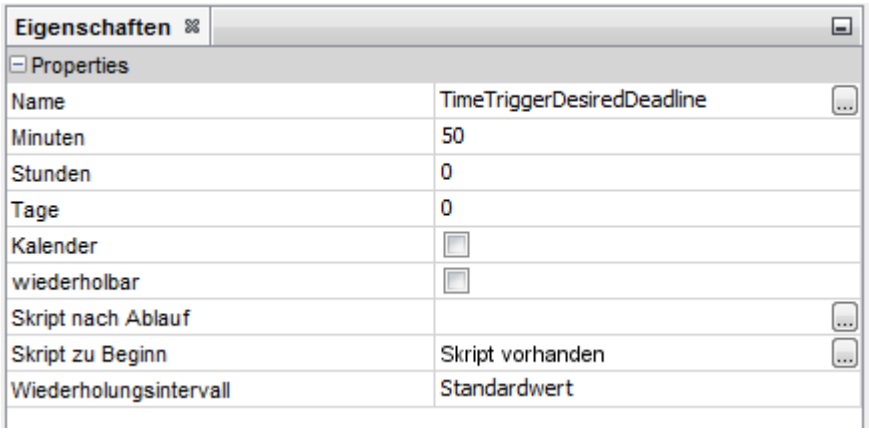


Abbildung 55: *ConSol CM Process Designer - Eigenschaften eines Zeit-Triggers*

C.7.1.4 Geschäftslogik und Initialisierung eines Zeit-Triggers

Die Zeitmessung eines Triggers wird begonnen (d. h. der Trigger wird initialisiert), wenn das Ticket in den Bereich bzw. die Aktivität eintritt. Sie hält an (d. h. der Trigger feuert), wenn der definierte Zeitraum, der als fester Wert gesetzt wurde (Minuten/Stunden/Tage), oder die manuell definierte Zeit verstrichen ist.

Wenn Sie als Workflow-Entwickler einen Trigger mit anderen Werten initialisieren möchten, muss dies mit Skripten erfolgen. Dieser Abschnitt enthält einige kurze Beispiele, eine detaillierte Erklärung zur Programmierung mit Zeit-Triggern in Workflows finden Sie im Abschnitt Arbeiten mit Kalendern und Zeiten. In diesen Kapiteln stehen auch Code-Beispiele.

- **Beispiel 1:**
  Die Reaktionszeit für ein Ticket soll anhand der Priorität berechnet werden. Im *Skript zu Beginn* werden die verschiedenen Reaktionszeiten verwendet (ein guter Weg, um dies zu implementieren, wäre über kundenspezifische System-Properties) und die Reaktionszeit wird berechnet. Danach wird der Trigger initialisiert, d. h. das Zeitintervall wird gesetzt.
- **Beispiel 2:**
  Wenn eine E-Mail für ein Ticket eingegangen ist und nach drei Stunden noch kein Bearbeiter die E-Mail gelesen und sich um das Ticket gekümmert hat, soll eine Warnung ausgelöst werden. Um dies zu implementieren, steht nach einer eingehenden E-Mail (siehe Abschnitt Mail-Trigger) eine automatische Aktivität, die den Zeit-Trigger mit drei Stunden reinitialisiert.

Ein Zeit-Trigger kann auch deaktiviert werden. In *Beispiel 2* ist dies erforderlich, um zu verhindern, dass der Zeit-Trigger am Anfang feuert, da er erst initialisiert werden soll, wenn eine E-Mail eingegangen ist.

C.7.1.5 Beispiele für Zeit-Trigger

Die Implementierungen für die oben genannten Anwendungsfälle (siehe Einführung in Zeit-Trigger) wären:

- **Anwendungsfall 1:**
  Hängen Sie einen manuellen Zeit-Trigger an die Aktivität *Wiedervorlage*. Der Bearbeiter kann das gewünschte Enddatum im Datumsauswahl-Fenster im Web Client auswählen. Normalerweise wird das Ticket dann wieder in die aktiven Tickets verschoben.

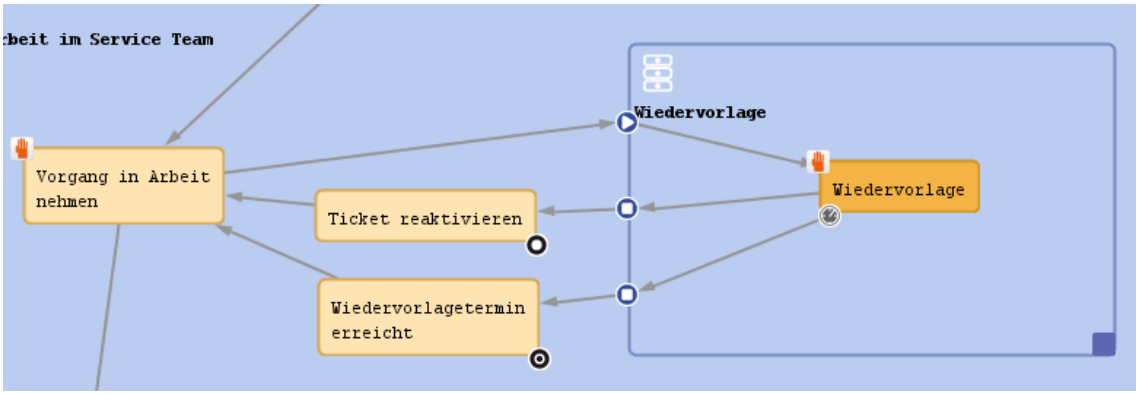


Abbildung 56: *ConSol CM Process Designer - Anwendungsfall 1: Workflow*

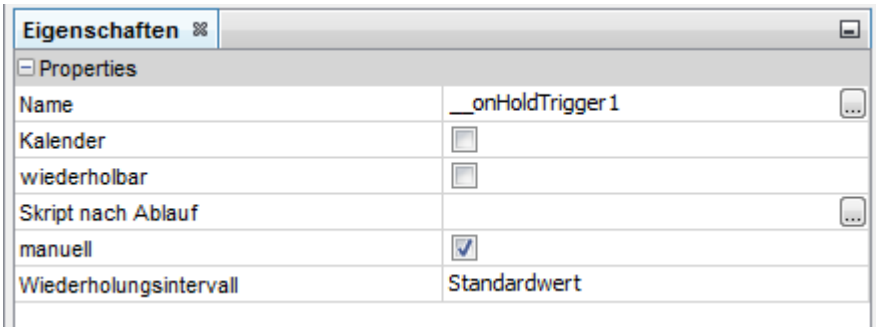


Abbildung 57: *ConSol CM Process Designer - Anwendungsfall 1: Eigenschaften-Editor für Zeit-Trigger*

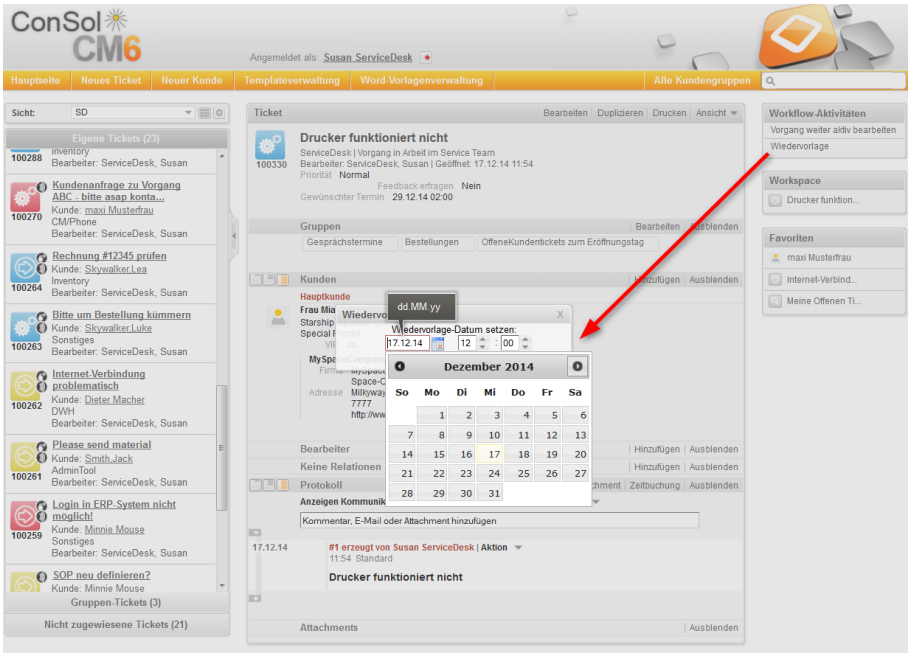


Abbildung 58: *ConSol CM Web Client - Anwendungsfall 1: Datumsauswahl-Feld*

- **Anwendungsfall 2:**
  Setzen Sie einen Zeit-Trigger an den Bereich, in dem neue Tickets erstellt werden. Definieren Sie die Zeit für den Trigger (kann von SLAs abhängen), z. B. vier Stunden. Setzen Sie einen Entscheidungsknoten hinter den Trigger, um festzustellen, ob das Ticket von einem Bearbeiter bearbeitet wird oder nicht. Wenn nicht, wird eine E-Mail an den Teamleiter gesendet.

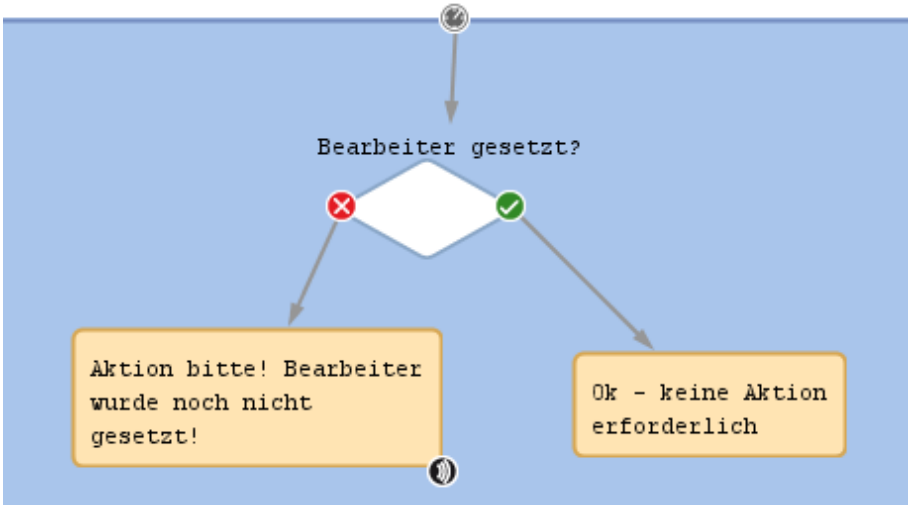


Abbildung 59: *ConSol CM Process Designer - Anwendungsfall 2: Workflow*

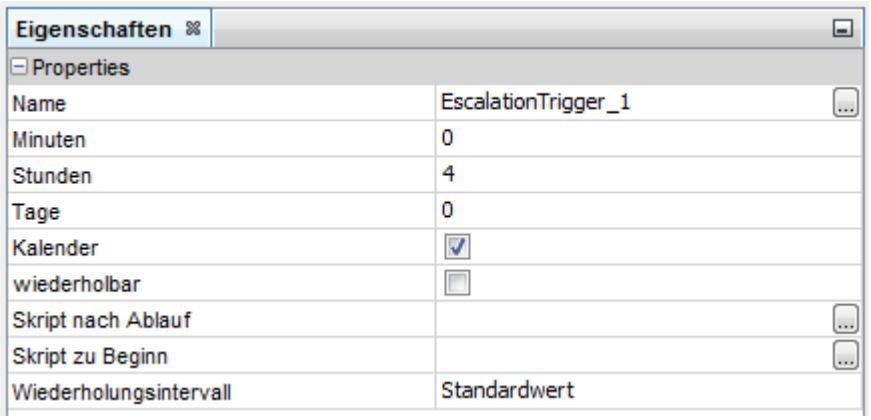


Abbildung 60: *ConSol CM Process Designer - Anwendungsfall 2: Eigenschaften-Editor für Zeit-Trigger*

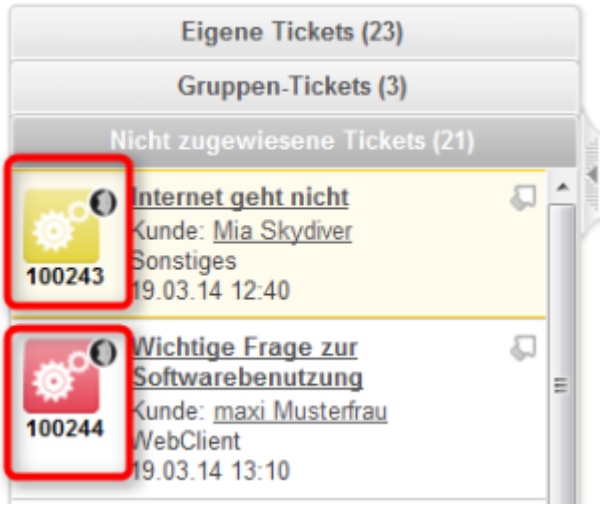


Abbildung 61: *ConSol CM Web Client - Anwendungsfall 2: Ticketliste*

- **Anwendungsfall 3:**
  Setzen Sie einen Zeit-Trigger an die Aktivität *Ticket abschließen mit Löschung (positiv)* und setzen Sie eine bestimmte Zeitspanne für den Trigger, z. B. fünf Tage. Hinter dem Trigger befindet sich der Endknoten des Prozesses. Das Ticket kann fünf Tage lang noch editiert werden. Danach wird es automatisch geschlossen.

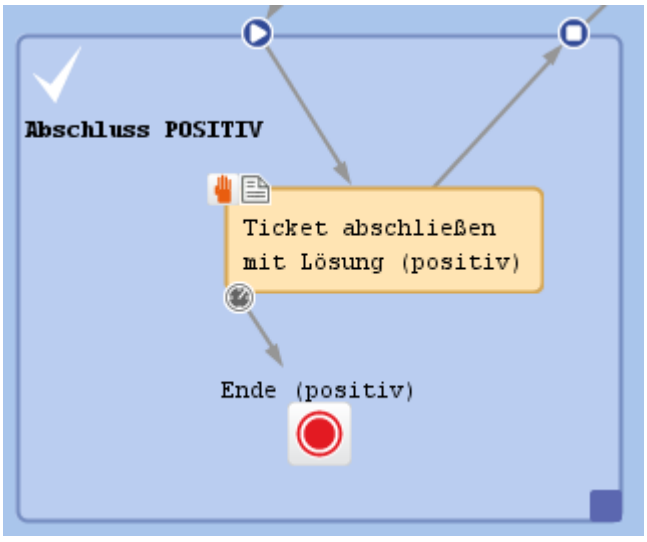


Abbildung 62: *ConSol CM Process Designer - Anwendungsfall 3: Workflow*

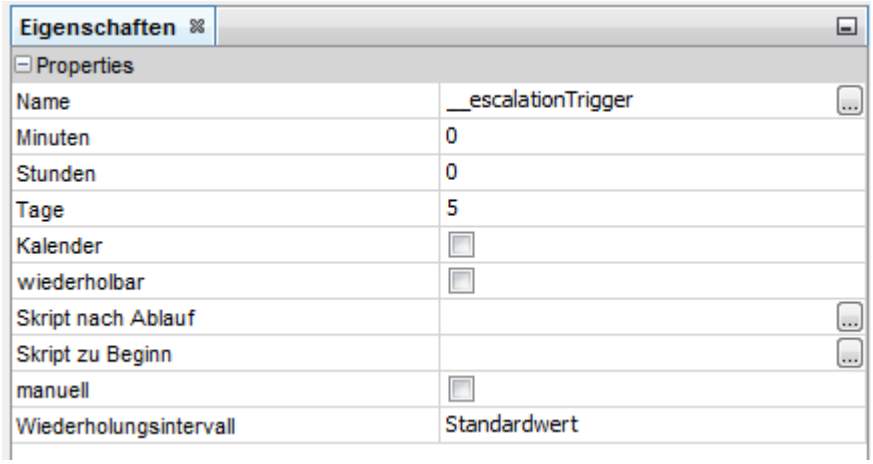


Abbildung 63: *ConSol CM Process Designer - Anwendungsfall 3: Eigenschaften-Editor für Zeit-Trigger*

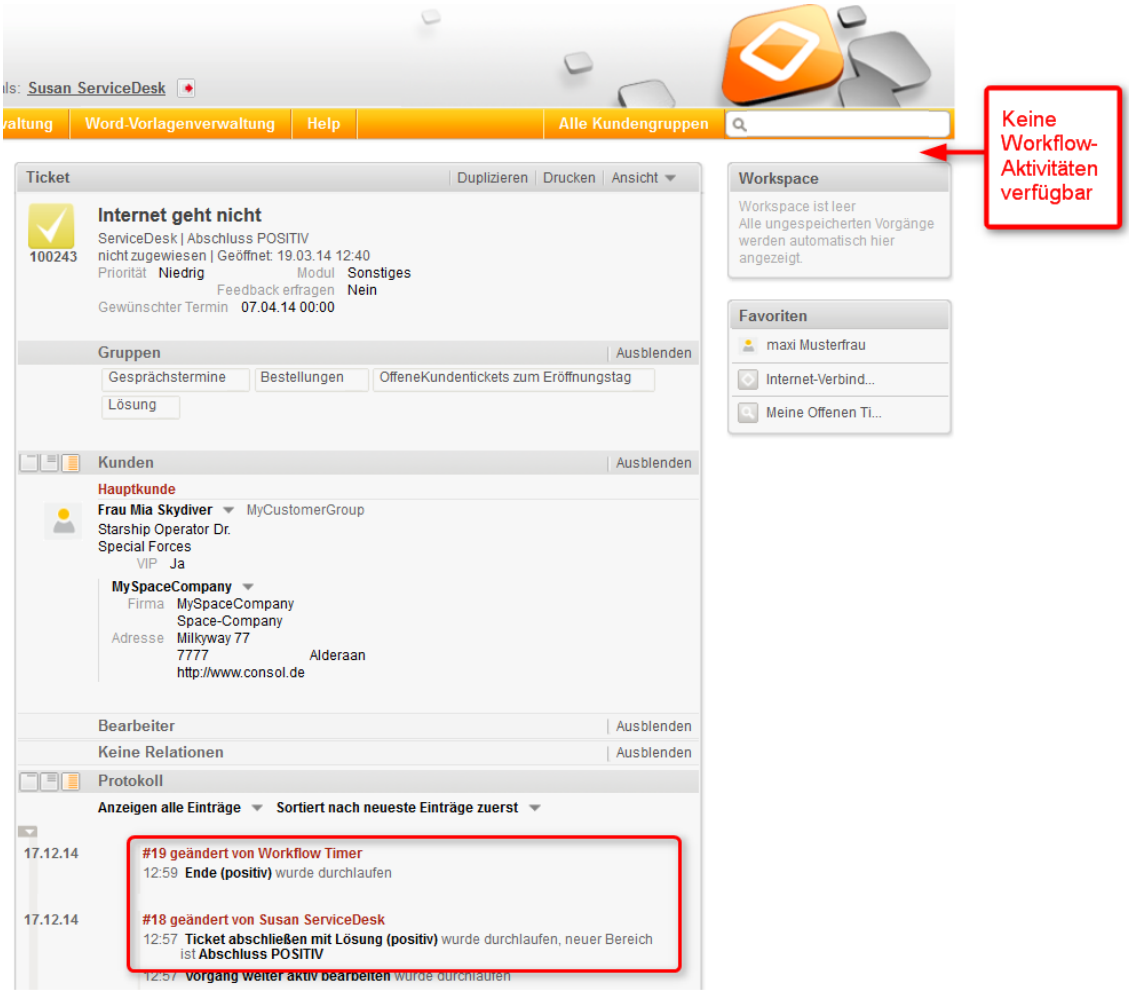


Abbildung 64: *ConSol CM Web Client - Geschlossenes Ticket*

C.7.1.6 Skripte mit Zeit-Triggern

Die folgenden Methoden sind wichtig, wenn Sie mit Zeit-Triggern arbeiten:

TimerTrigger

Das wichtigste Objekt in einem Skript für einen Trigger ist der Trigger selber. Es handelt sich um ein Objekt der Java-Klasse *TimerTrigger* und es ist implizit in jedem Trigger-Skript als *trigger* verfügbar.

- **TimerTrigger.setDueTime(long pDueTime in millisecs)**
  Setzt die Zeit, nach der der Trigger feuern soll, in Millisekunden angegeben. Die als Methodenparameter angegebene Zeit wird zur ursprünglichen Startzeit des Triggers hinzugerechnet, d. h. zu dem Zeitpunkt, an dem das Ticket in den Bereich bzw. die Aktivität gewandert ist, in der sich der Trigger befindet. *setDueTime()* definiert also den Zeitraum in Millisekunden von der Eintrittszeit zum gewünschten Feuern des Triggers.

workflowApi

workflowApi (die Singleton-Instanz von WorkflowContextService) bietet zwei Methoden zum Reinitialisieren des Triggers. Reinitialisieren bedeutet, dass der Trigger auf seinen ursprünglichen Zustand zurückgesetzt wird, sodass noch keine Zeit abgelaufen ist. Für beide Methoden muss der Name des Triggers (pTriggerName) als Pfad angegeben werden. Eine Erklärung dazu finden Sie im Abschnitt über die Arbeit mit Pfadinformationen.

- **reinitializeTrigger(String pTriggerName)**
  Der Trigger wird mit dem Startdatum, an dem das Ticket in den entsprechenden Bereich gewandert ist, reinitialisiert.
- **reinitializeTrigger(String pTriggerName, Date pBaseDate)**
  Der Trigger wird reinitialisiert und das Startdatum (BaseDate) wird explizit gesetzt. Auf diese Weise kann ein Trigger mit einem Datum reinitialisiert werden, das vom Datum abweicht, an dem das Ticket in den entsprechenden Bereich gewandert ist. pBaseDate ist ein absolutes Datum, das als Java-Datumsobjekt angegeben ist.
- **workflowApi.deactivateTimer()**
  (unterschiedliche Methodensignaturen)
  Deaktiviert den angegebenen Zeit-Trigger, d. h. der Trigger feuert erst, wenn er reinitialisiert wird.
  (Es gibt **keine** Methode *activateTimer().* Verwenden Sie *workflowApi.reinitializeTrigger()*, um einen Trigger wieder zu aktivieren).

Siehe auch Abschnitt Arbeiten mit Kalendern und Zeiten.

Beispiel 1: Setzen der Ablaufzeit eines Zeit-Triggers entsprechend der Queue

Diese Skript könnte beim Start der Zeitmessung für einen Zeit-Trigger an einem Bereich verwendet werden. Es initialisiert den Trigger für eine Eskalation entsprechend der Queue, d. h. wenn sich das Ticket in der Queue *HelpDesk_1st_Level* befindet, erfolgt die Eskalation schneller als in der Queue *HelpDesk_2nd_Level*.

Innerhalb der Skripte *Skript zu Beginn* und *Skript nach Ablauf* existiert das Objekt *trigger* als implizite Initialisierung von *TimerTrigger*. Sie können also direkt mit dem Trigger arbeiten, ohne vorher zusätzliche Schritte durchführen zu müssen. In einem Admin-Tool-Skript müssen Sie allerdings die Klasse *TimerTrigger* oder das entsprechende Java-Paket vorher importieren.

Das folgende Skript kann in einer Servicedesk- und Helpdesk-Umgebung verwendet werden und an den folgenden *TimerTrigger* angehängt werden.

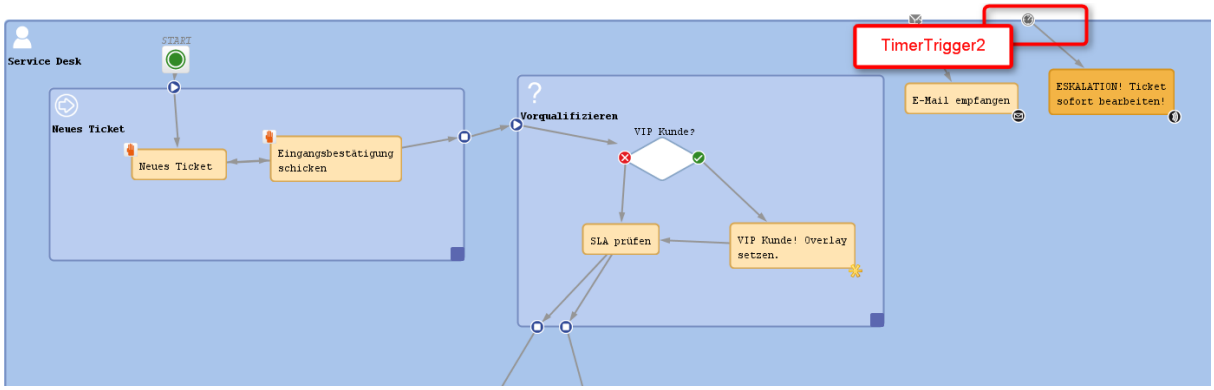


Abbildung 65: *ConSol CM Process Designer - TimerTrigger im ServiceDesk-Workflow*

```
def addedEscalMillis = 0
switch (ticket.queue.name) {
   case "HelpDesk_1st_Level":
      addedEscalMillis = 12*60*60*1000L;
      break;
   case "HelpDesk_2nd_Level":
      addedEscalMillis = 24*60*60*1000L;
      break;
   case "ServiceDesk":
      addedEscalMillis = 4*60*60*1000L;
   }
trigger.setDueTime(addedEscalMillis)
```

Code-Beispiel 7: *Beispiel für ein Skript zu Beginn*

> ⚠ Für dieses Beispiel ist es sinnvoll, statische Werte für die Zeiten direkt im Skriptcode anzugeben. In echten Umgebungen empfiehlt es sich, die Eskalationszeiten und dergleichen in System-Properties zu speichern und über *configurationService* abzurufen. Auf diese Weise kann ein Administrator einfach auf die Eskalationszeiten zugreifen und diese ändern, ohne die Workflow-Implementierung verändern zu müssen.
>
> Im Arbeitsalltag würde vielleicht auch ein Arbeitszeitkalender verwendet werden, siehe *Beispiel 2*.

In der Datei *server.log* sehen Sie die Zeit, zu der der Trigger feuern soll.

Abbildung 66: *Datei server.log mit berechneter Ablaufzeit des Zeit-Triggers*

Das gleiche Prinzip wird angewendet, um die Eskalationszeit abhängig von der Ticketpriorität, dem *VIP-Status* des Kunden oder einem anderen Parameter zu berechnen.

Beispiel 2: Eskalation als Warnung 2 Tage vor dem gewünschten Enddatum berechnen

```
def now = new Date()
def wunschTermin = ticket.get("helpdesk_standard", "date_test")
def twoWorkDays = -2*8*60*60*1000L

// calculate escalation date
def escalDate = BusinessCalendarUtil.getBusinessTime(wunschTermin, twoWorkDays,
 ticket.queue.calendar)
// calculate and set due time
trigger.setDueTime(escalDate.time - now.time)
```

Code-Beispiel 8: *Zeit für TimerTrigger mit BusinessCalendar berechnen und setzen*

C.7.2 Mail-Trigger

In diesem Kapitel werden folgende Themen behandelt:

- Einführung in Mail-Trigger
- Hinzufügen eines Mail-Triggers zu einem Workflow
- Eigenschaften eines Mail-Triggers
- Beispiele für Mail-Trigger
- Prozesslogik mit Mail-Triggern

C.7.2.1 Einführung in Mail-Trigger

Eine der Kernfunktionalitäten von ConSol CM ist das Zusammenspiel mit einer E-Mail-Infrastruktur. Auf diese Weise können Bearbeiter manuelle E-Mails senden, und das System kann automatische E-Mails an Kunden und Bearbeiter senden, so wie es im entsprechenden Prozessschritt erforderlich ist. Natürlich muss ConSol CM auch E-Mails empfangen. Dazu werden E-Mails aus einem oder mehreren Postfächern mit ConSol CM-E-Mail-Adressen abgerufen. Eine detaillierte Erklärung des Zusammenspiels zwischen dem Mail-Server und ConSol CM finden Sie im *ConSol CM Administratorhandbuch* und *ConSol CM Betriebshandbuch*. An dieser Stelle ist nur das Zusammenspiel mit dem Workflow erklärt.

Abbildung 67: *ConSol CM Process Designer - Mail-Trigger*

Mail-Trigger an einem Bereich

Wenn eine E-Mail empfangen wird, die zu einem vorhandenen und aktiven (offenen) Ticket gehört, ist es möglicherweise erforderlich, diese Aktion zu erfassen und danach bestimmte Aktionen durchzuführen. Dies kann durch die Verwendung von einem oder mehreren Mail-Triggern in einem Workflow erreicht werden.

> ⚠ Bedenken Sie, dass (in der Standardkonfiguration, d. h. ohne Änderung des Admin-Tool-Skripts *AppendToTicket.groovy*) das Anhängen der E-Mail an das Ticket über den Ticket-Tag im Betreff, z. B. *Ticket (<TicketNumber>)*, die **einzige automatische Aktion** ist, die von ConSol CM nach Erhalt einer E-Mail in einem bestimmten Postfach durchgeführt wird. Siehe auch *ConSol CM Administratorhandbuch*, Abschnitt *Skripte des Typs E-Mail*.
>
> **Alle anderen Aktionen**, die durchgeführt werden sollen, wenn eine E-Mail eingeht, müssen **manuell** im Workflow (und/oder in Admin-Tool-Skripten) programmiert werden!

Beispiele für die Verwendung von Mail-Triggern sind:

Wenn eine E-Mail eingeht ...

- soll der zugewiesene Bearbeiter des Tickets eine E-Mail als Benachrichtigung erhalten.
- soll das Ticket-Icon (im Web Client) durch ein Overlay gekennzeichnet werden.
- soll das Ticket an eine Aktivität übergeben werden, in der der Bearbeiter bestätigen muss, dass er die E-Mail gelesen hat.
- sollen der Absender und der Betreff der E-Mail überprüft und verarbeitet werden. Wenn es sich bei der E-Mail um eine Bestätigung oder Ablehnung in einem Genehmigungsprozess handelt, wird das Ticket entsprechend der im Workflow definierten Regeln und Aktivitäten behandelt. Auf diese Weise kann die Genehmigung direkt über E-Mail erfolgen und der Genehmiger muss sich nicht im Web Client anmelden.

Mail-Trigger an einer Aktivität

Wenn ein Mail-Trigger an einer Aktivität hängt, wird diese Aktivität nur ausgeführt, wenn eine E-Mail eingegangen ist.

Abbildung 68: *ConSol CM Process Designer - Mail-Trigger an Aktivität*

C.7.2.2 Hinzufügen eines Mail-Triggers zu einem Workflow

Hinzufügen eines Mail-Triggers zu einem Bereich

Klicken Sie in der Palette auf das Mail-Trigger-Symbol und ziehen Sie es in den gewünschten Bereich. Es wird automatisch an den oberen Rand des Bereichs angefügt. Sie können die Position später ändern (ziehen Sie den Trigger nach links oder rechts, um das Layout zu verbessern). Es kann pro Bereich nur ein Mail-Trigger verwendet werden.

Ein Mail-Trigger, der an einem Bereich hängt, kann nicht in einen anderen Bereich verschoben werden. Wenn Sie einen Mail-Trigger an einen anderen Bereich anfügen möchten, entfernen Sie den bereits erstellen Trigger und erstellen Sie einen neuen Trigger für den richtigen Bereich.

Sie können vom Trigger ausgehend Verbindungen zu Aktivitäten oder Entscheidungsknoten ziehen, die sich hinter dem Trigger befinden sollen. Der erste Schritt, der nach dem Mail-Trigger ausgeführt wird, muss immer eine automatische Aktivität sein!

Hinzufügen eines Mail-Triggers zu einer Aktivität

In dem seltenen Fall, dass Sie einen Mail-Trigger an eine Aktivität hängen müssen (wir empfehlen dies nicht!), klicken Sie in der Palette auf das Mail-Trigger-Symbol und ziehen es in die gewünschte Aktivität. Es wird an die Ecke der Aktivität angefügt.

Ein Mail-Trigger, der an einer Aktivität hängt, kann nicht in eine anderen Aktivität oder einen Bereich verschoben werden. Wenn Sie einen Mail-Trigger an eine andere Aktivität oder einen Bereich anfügen möchten, entfernen Sie den bereits erstellen Trigger und erstellen Sie einen neuen Trigger für die richtige Aktivität bzw. den Bereich.

C.7.2.3 Eigenschaften eines Mail-Triggers

Ein Mail-Trigger hat keine Eigenschaften.

C.7.2.4 Beispiele für Mail-Trigger

Anwendungsfall 1: Overlay für Ticket-Icon

Wenn für ein Ticket, das sich gerade in dem Bereich befindet, eine E-Mail eingegangen ist, soll das Ticket-Icon im Web Client mit dem Overlay *mail* gekennzeichnet werden.

Der Mail-Trigger hängt am Bereich und das Overlay wird in der nachfolgenden automatischen Aktivität, die mit dem Trigger verbunden ist, hinzugefügt. Die Overlay-Gültigkeit ist *Aktivität*.

Auf diese Weise wird das Ticket mit dem Overlay gekennzeichnet, wenn eine E-Mail eingegangen ist. Sobald der Bearbeiter das Ticket in eine andere Aktivität bewegt hat, verschwindet das Overlay.

Bedenken Sie, dass das Ticket seinen Kontext nicht verlässt. Das einzige, was passiert, ist, dass ein Overlay zum Ticket-Icon hinzugefügt wird. Das Ticket kehrt dann an seine ursprüngliche Position im Workflow zurück. Dies wird auch Interrupt genannt. Eine detaillierte Erklärung dazu finden Sie im Abschnitt Prozesslogik.

Abbildung 69: *ConSol CM Process Designer - Anwendungsfall 1: Bereich mit Mail-Trigger*

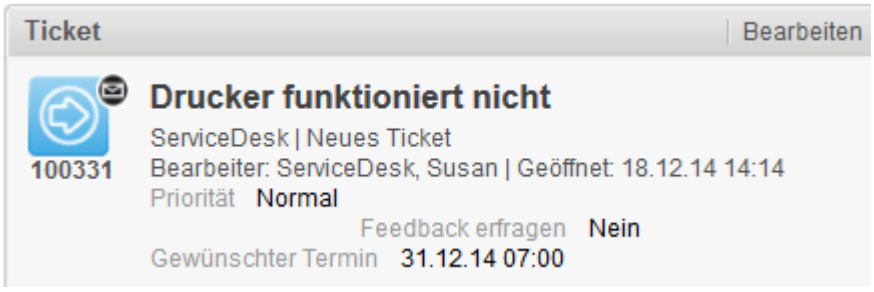


Abbildung 70: *ConSol CM Web Client - Anwendungsfall 1: Ticket mit Overlay-Icon*

Anwendungsfall 2: Overlay für das Ticket-Icon und E-Mail-Bestätigung durch den Bearbeiter

Wenn für ein Ticket, das sich gerade in dem Bereich befindet, eine E-Mail eingegangen ist, soll das Ticket-Icon im Web Client mit dem Overlay *mail* gekennzeichnet werden. Zusätzlich soll das Ticket an eine Position verschoben werden, an der es wartet, bis der Bearbeiter bestätigt, dass er die E-Mail gelesen hat.

Der Mail-Trigger hängt am Bereich und das Overlay wird in der nachfolgenden automatischen Aktivität hinzugefügt. Die Overlay-Gültigkeit ist *Aktivität*. Auf diese Weise wird das Ticket mit dem Overlay gekennzeichnet, wenn eine E-Mail eingegangen ist.

Innerhalb des Skripts, das auf den Mail-Trigger folgt, wird das Boolean-Feld *mail_to_read* auf *true* gesetzt. Im Workflow wird bei Bedarf die Aktivität *E-Mail gelesen* angeboten. Sie wird nur angezeigt, wenn der Wert des Boolean-Feldes *mail_to_read true* ist. Dies ist ein stärkerer Mechanismus, um den Bearbeiter an eine eingegangene E-Mail zu erinnern, als nur ein Overlay. Der Bearbeiter muss die E-Mail bestätigen, indem er die Workflow-Aktivität *E-Mail gelesen* explizit ausführt. Innerhalb dieser Workflow-Aktivität wird der Wert des Boolean-Feldes *mail_to_read* wieder auf *false* gesetzt. Jetzt ist das Ticket bereit, eine weitere E-Mail zu erhalten und den Bearbeiter zu benachrichtigen.

Bedenken Sie auch hier, dass das Ticket seinen Kontext nicht in Folge der Aktion verlässt, die nach Eingang der E-Mail ausgeführt wird. Es wird lediglich ein Overlay an das Ticket-Icon angefügt und eine Variable des Typs *boolean* geändert. Das Ticket kehrt dann an seine ursprüngliche Position im Workflow zurück. Es handelt sich also um einen Interrupt. Eine detaillierte Erklärung dazu finden Sie im Abschnitt Prozesslogik.

Abbildung 71: *ConSol CM Process Designer - Anwendungsfall 2: Bereich mit Mail-Trigger*

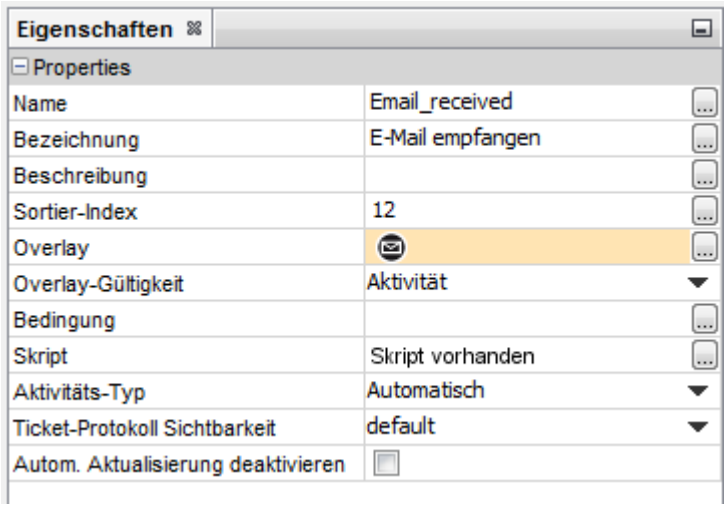


Abbildung 72: *ConSol CM Process Designer - Anwendungsfall 2: Eigenschaften der Aktivität "E-Mail empfangen"*

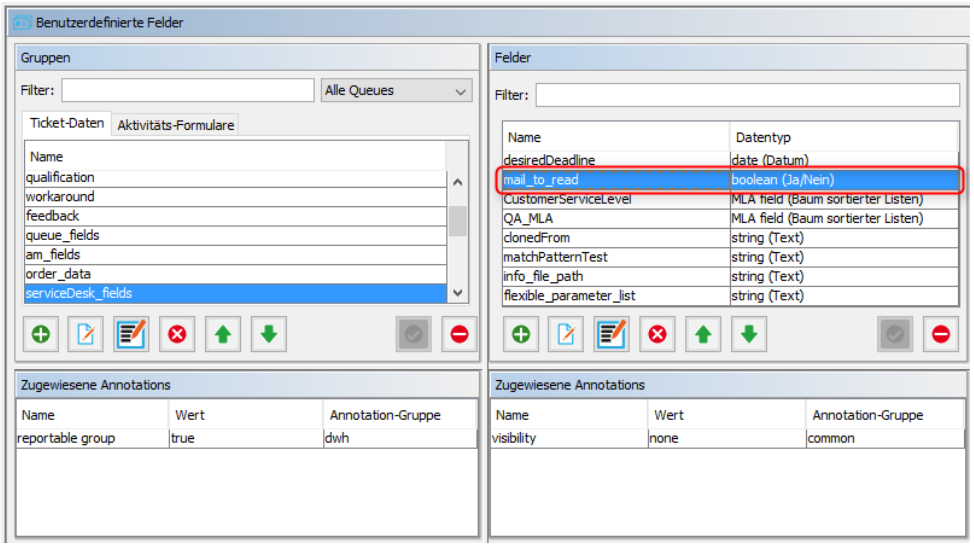


Abbildung 73: *ConSol CM Admin Tool - Anwendungsfall 2: Neues Boolean-Feld zum Erfassen der E-Mail*

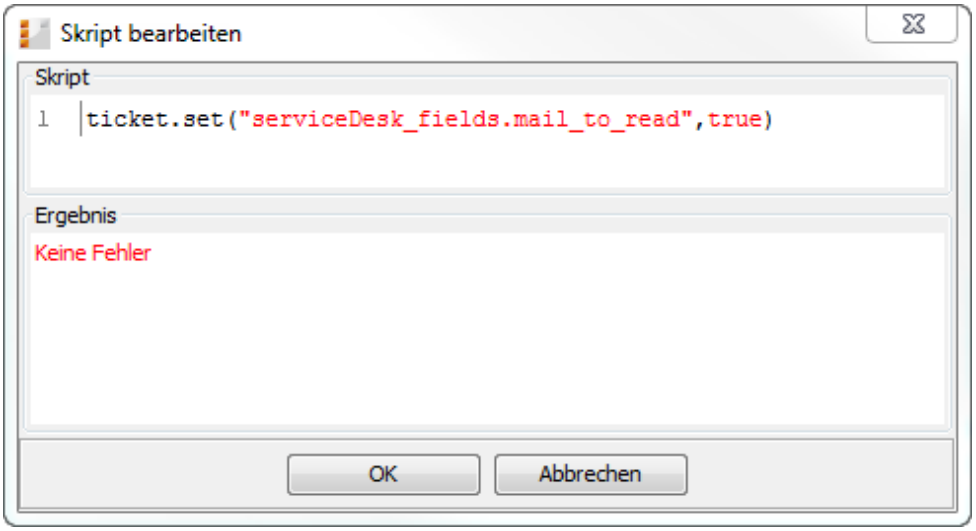


Abbildung 74: *ConSol CM Process Designer - Anwendungsfall 2: Skript der Aktivität "E-Mail empfangen"*

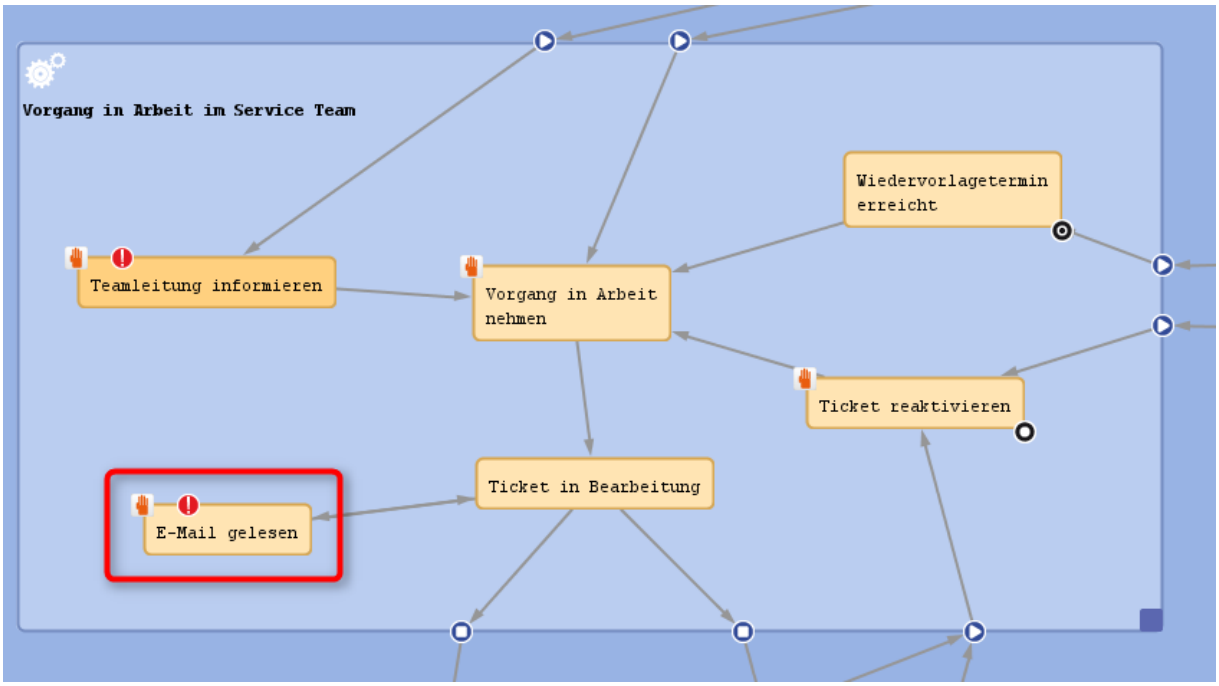


Abbildung 75: *ConSol CM Process Designer - Anwendungsfall 2: Aktivität "E-Mail gelesen"*

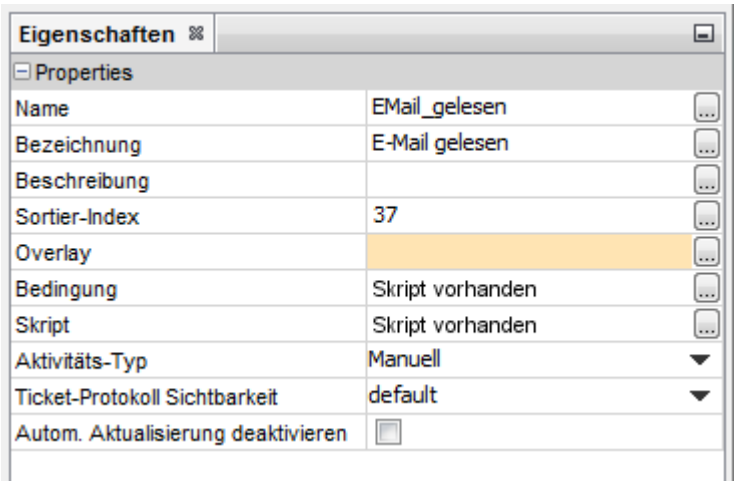


Abbildung 76: *ConSol CM Process Designer - Anwendungsfall 2: Eigenschaften der Aktivität "E-Mail gelesen"*

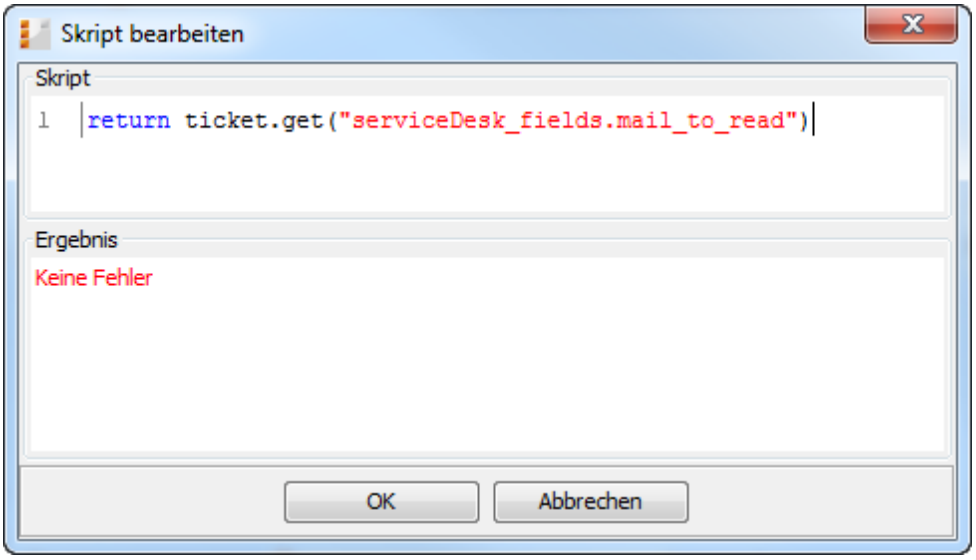


Abbildung 77: *ConSol CM Process Designer - Anwendungsfall 2: Bedingungsskript der Aktivität "E-Mail gelesen"*

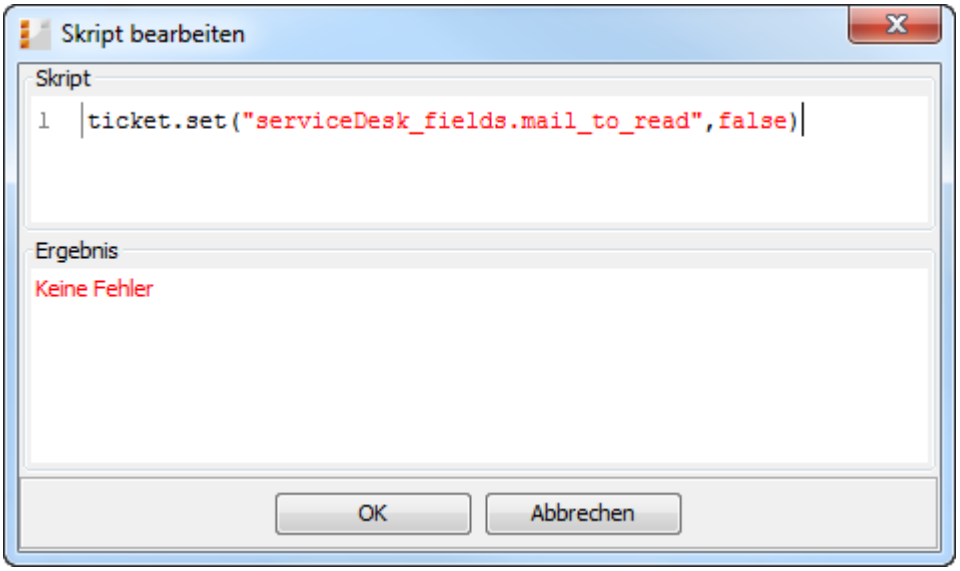


Abbildung 78: *ConSol CM Process Designer - Anwendungsfall 2: Skript der Aktivität "E-Mail gelesen"*

Abbildung 79: *ConSol CM Web Client - Anwendungsfall 2: Workflow-Aktivität "E-Mail gelesen"*

C.7.2.5 Prozesslogik mit Mail-Triggern

Wenn eine E-Mail eingegangen ist, feuert zuerst der Mail-Trigger des innersten der möglichen Bereiche.

Beispiel 1:
Das Ticket befindet sich an Position **(1)** im Bereich *Wiedervorlage*. Wenn eine E-Mail eingeht, feuert zuerst der Mail-Trigger dieses Bereichs **(2)** und infolgedessen wird das Ticket in einen anderen Bereich verschoben **(3)**.

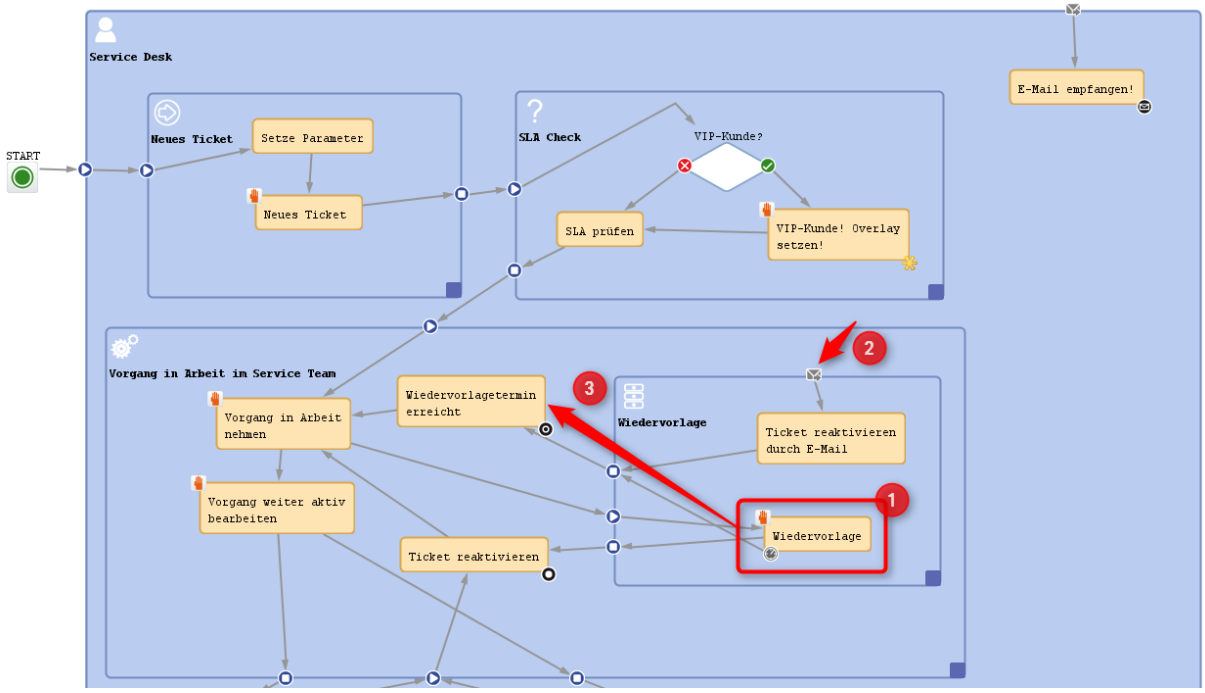


Abbildung 80: *ConSol CM Process Designer - Beispiel 1: Mail-Trigger eines Unterbereichs aktiv*

Beispiel 2:
Das Ticket befindet sich an Position **(1)** im Bereich *Vorgang in Arbeit*. Wenn eine E-Mail eingeht, feuert der Mail-Trigger des globalen Bereichs **(2)** (da der Bereich *Vorgang in Arbeit* keinen Mail-Trigger hat). Die Position des Tickets wird also nicht verändert **(3)**.

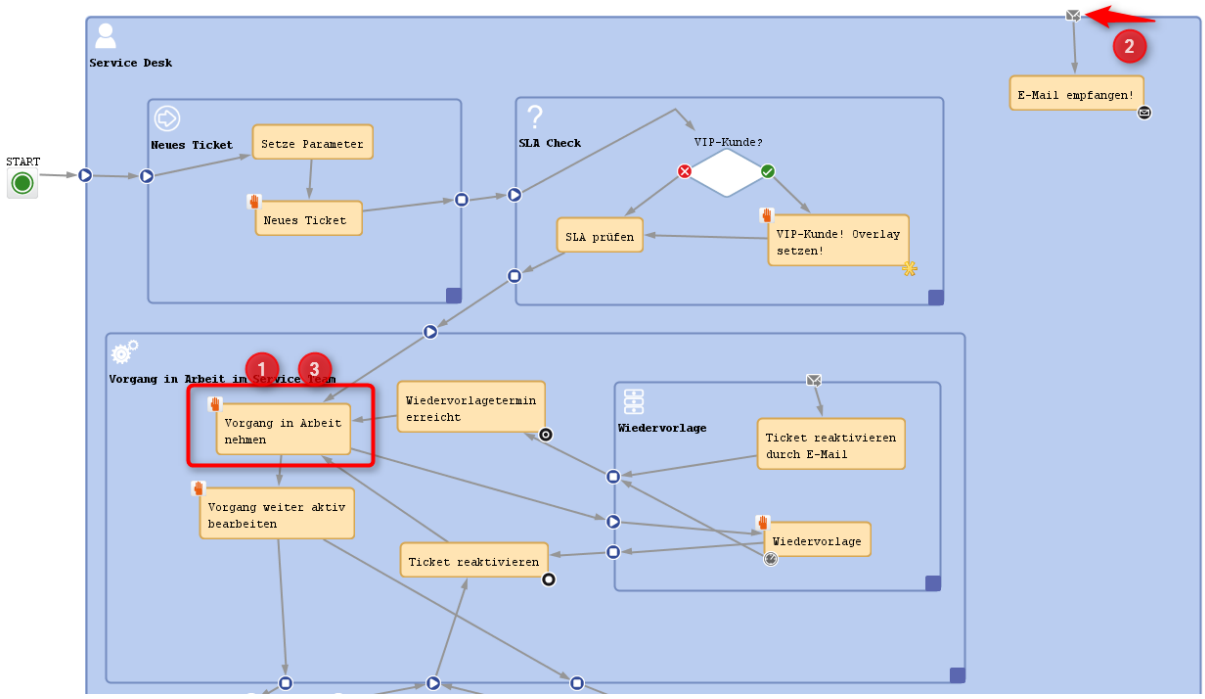


Abbildung 81: *ConSol CM Process Designer - Beispiel 2: Mail-Trigger des globalen Bereichs aktiv*

C.7.3 Event-Trigger

In diesem Kapitel werden folgende Themen behandelt:

- Einführung zu Event-Tiggern
- Hinzufügen eines Event-Triggers zu einem Workflow
- Eigenschaften eines Event-Triggers
- Geschäftslogik von Event-Triggern
- Beispiele für Event-Trigger
- Best Practices: Verwendung von Event-Triggern

C.7.3.1 Einführung zu Event-Tiggern

In Geschäftsprozessen treten häufig Ereignisse auf, um die sich ein Bearbeiter im Rahmen des regulären Prozesses kümmern muss. Zum Beispiel kann es erforderlich sein, den Teamleiter zu informieren, wenn jemand die Priorität eines Tickets auf *Sehr hoch* stellt. Oder es muss nach dem Zuweisen eines neuen Bearbeiters zu einem Ticket überprüft werden, ob dieser Bearbeiter angemeldet ist (wenn er nicht angemeldet ist, muss das Ticket an einen anderen Bearbeiter übertragen werden). Es gibt im Arbeitsalltag eine Vielzahl an Beispielen für solche Ereignisse.

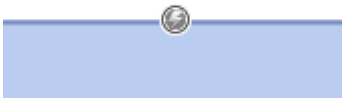


Abbildung 82: *ConSol CM Process Designer - Event-Trigger*

ConSol CM kann Ereignisse mithilfe von Event-Triggern erfassen und auf die folgenden Arten von Ereignissen reagieren:

- Ändern des Bearbeiters
- Ändern der Queue
- Ändern des Themas
- Ändern von zusätzlichen Bearbeitern
- Verknüpfte Ressourcen
- Zeitbuchung
- Ändern eines Kommentars
  (normalerweise durch Hinzufügen eines Kommentars, d. h. eines Textkommentars oder einer E-Mail)
- Ändern eines Benutzerdefinierten Feldes, das vom Systementwickler definiert wird
  (dies kann z. B. die Priorität, eine Kategorie oder der Inhalt eines bestimmten Textfeldes sein)

Wenn das Ereignis eintritt, feuert der Event-Trigger.

Sie als Workflow-Entwickler müssen alles implementieren, was geschehen soll, wenn ein Event-Trigger gefeuert hat! Es gibt keine automatischen Aktionen. Der Event-Trigger gibt lediglich das Signal *Ereignis eingetreten*.

Ein Workflow kann so viele Event-Trigger haben, wie benötigt werden. Sie müssen allerdings sicherstellen, dass es im Prozess möglich ist, dass alle Event-Trigger potentiell feuern können (und dass keiner von einer Aktion abhängt, die nie auftreten kann, da ein anderer Event-Trigger (oder Zeit-Trigger) vorher gefeuert hat). Weitere Informationen dazu finden Sie im Abschnitt Prozesslogik.

C.7.3.2 Hinzufügen eines Event-Triggers zu einem Workflow

Event-Trigger können nur an Bereichen hängen, nie an Aktivitäten.

Hinzufügen eines Event-Triggers zu einem Bereich

Klicken Sie in der Palette auf das Event-Trigger-Symbol und ziehen Sie es in den gewünschten Bereich. Es wird automatisch an den oberen Rand des Bereichs angefügt. Sie können die Position später ändern (ziehen Sie den Trigger nach links oder rechts, um die Reihenfolge der Trigger zu ändern oder das Layout zu verbessern).

Ein Event-Trigger, der an einem Bereich hängt, kann nicht in einen anderen Bereich verschoben werden. Wenn Sie einen Event-Trigger an einen anderen Bereich anfügen möchten, entfernen Sie den bereits erstellten Trigger und erstellen Sie einen neuen Trigger für den richtigen Bereich.

Um die Eigenschaften des Triggers zu konfigurieren, wählen Sie ihn im Bearbeitungsbereich aus und setzen Sie im Eigenschaften-Editor die richtigen Werte. Siehe Abschnitt Eigenschaften eines Event-Triggers.

Sie können vom Trigger ausgehend Verbindungen zu Aktivitäten oder Entscheidungsknoten ziehen, die sich hinter dem Trigger befinden sollen. Der erste Schritt, der nach dem Event-Trigger ausgeführt wird, muss immer eine automatische Aktivität sein!

C.7.3.3 Eigenschaften eines Event-Triggers

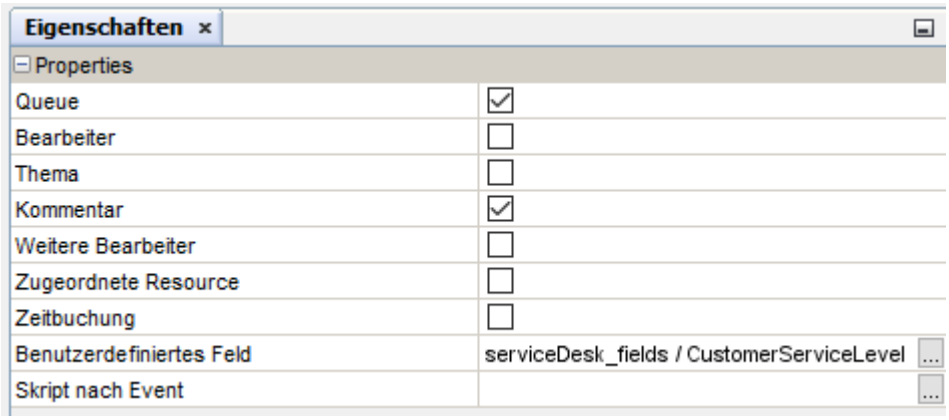


Abbildung 83: *ConSol CM Process Designer - Eigenschaften eines Event-Triggers*

Ein Event-Trigger kann folgende Eigenschaften haben:

- **Queue**
  Markieren Sie diese Checkbox, wenn der Event-Trigger auf den Wechsel der Queue reagieren soll, d. h. der Trigger feuert, wenn ein Ticket in eine andere Queue verschoben wird. Es ist nicht relevant, ob es sich um eine manuelle Aktion oder eine automatisch vom System durchgeführte Aktion handelt.
- **Bearbeiter**
  Markieren Sie diese Checkbox, wenn der Trigger auf den Wechsel des zugewiesenen Bearbeiters des Tickets reagieren soll. Dies kann eine manuelle oder eine automatische Aktion sein. Es gibt drei mögliche Konstellationen:
  - Das Ticket hatte keinen Bearbeiter und es wird ein Bearbeiter gesetzt.
  - Das Ticket hat einen Bearbeiter und wird einem anderen Bearbeiter zugewiesen.
  - Das Ticket ist einem Bearbeiter zugewiesen und der Bearbeiter wird auf *null* (kein Bearbeiter) gesetzt.
- **Thema**
  Markieren Sie diese Checkbox, wenn der Trigger auf eine Änderung des Ticketthemas reagieren soll.
- **Kommentar**
  Markieren Sie diese Checkbox, wenn der Trigger auf eine Änderung eines Kommentars reagieren soll, d. h.:
  - Ein Bearbeiter hat einen neuen Kommentar (Text) eingegeben.
  - Ein Kunde hat über ConSol CM.Track einen neuen Kommentar (Text) eingegeben.
  - Das Ticket hat eine E-Mail erhalten.
  - Aus dem Ticket wurde eine E-Mail gesendet.
  - Es wurden ein oder mehrere Attachments zum Ticket hinzugefügt.
- **Weitere Bearbeiter**
  Markieren Sie diese Checkbox, wenn der Trigger auf eine Änderung von zusätzlichen Bearbeitern mit bestimmten Bearbeiterrollen im Ticket reagieren soll (Ticketbereich *Zusätzliche Bearbeiter*). Dies kann in folgenden Situationen geschehen (manuell gesetzt oder automatisch vom System gesetzt):
  - Das Ticket hatte keine zusätzlichen Bearbeiter und es werden ein oder mehrere zusätzliche Bearbeiter gesetzt.
  - Das Ticket hat einen oder mehrere zusätzliche Bearbeiter und einer oder mehrere dieser Bearbeiter werden auf *null* gesetzt oder ihr Name wird geändert.
  - Das Ticket hat einen oder mehrere zusätzliche Bearbeiter und alle diese Bearbeiter werden auf *null* (kein Bearbeiter) gesetzt.
- **Zugeordnete Ressource**
  Markieren Sie diese Checkbox, wenn der Trigger auf die Änderung einer verknüpften Ressource im Ticket reagieren soll, d. h.:

- Die Relation zur Ressource wurde gelöscht.
- Es wurde eine neue Relation zu einer Ressource hergestellt.
- Eine Relation wird geändert (z. B. die Beschreibung wird geändert)

- **Zeitbuchung**
  Markieren Sie diese Checkbox, wenn der Trigger auf Änderungen an Zeitbuchungen im Ticket reagieren soll, d. h.:
  - Es wird eine neue Zeitbuchung hinzugefügt.
- **Benutzerdefiniertes Feld**
  Öffnen Sie das Pop-up-Fenster *Event Trigger* (siehe folgende Abbildung) mit dem Button (...) und wählen Sie die Benutzerdefinierten Felder, die überwacht werden sollen. Mit den Buttons *Plus* und *Minus* können Sie weitere Felder hinzufügen oder die Anzahl der überwachten Felder verringern. Wie bei der Definition von Benutzerdefinierten Feldern (siehe *ConSol CM Administratorhandbuch*, Abschnitt *Verwaltung von Benutzerdefinierten Feldern*) müssen Sie zuerst im linken Drop-down-Menü die Benutzerdefinierte Feldgruppe auswählen und dann im rechten Drop-down-Menü eines der Benutzerdefinierten Felder dieser Gruppe auswählen. Sie können so viele Benutzerdefinierte Felder auswählen, wie Sie möchten.

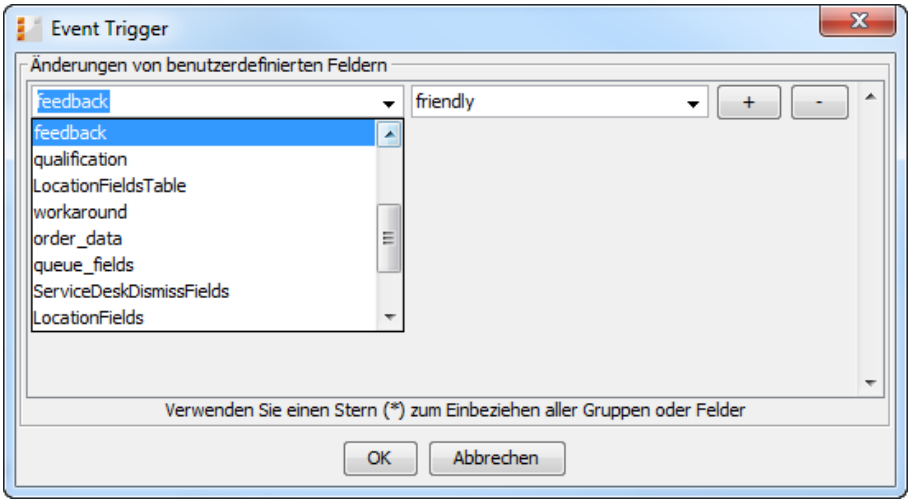


Abbildung 84: *ConSol CM Process Designer - Eigenschaft "Benutzerdefiniertes Feld" eines Event-Triggers*

- **Skript nach Event**
  Hier können Sie (mit dem ConSol CM-Skripteditor) ein Skript definieren, das ausgeführt werden soll, wenn der Event-Trigger gefeuert hat. Das Skript muss *true* oder *false* zurückgeben. Wenn es *true* zurückgibt, wird das Ereignis tatsächlich gefeuert, d. h. die automatische Aktivität nach dem Event-Trigger wird ausgeführt. Wenn das Skript *false* zurückgibt, wird das Ereignis blockiert und die automatische Aktivität wird nicht ausgeführt. Auf diese Weise können Sie genau steuern, wann die Aktion (Aktivität) ausgeführt werden soll, z. B. reagiert der Trigger auf eine Änderung der Priorität, feuert aber nur, wenn die neue Priorität *Sehr hoch* ist. Das Skript überprüft die neue Priorität und gibt nur dann *true* zurück, wenn der neue Wert *Sehr hoch* ist. Für alle anderen Werte gibt das Skript *false* zurück.

> ⚠ Das *Skript nach Event* wird nur verwendet, um das Feuern des Event-Triggers zu steuern und einzustellen! Alle Aktionen, die durchgeführt werden sollen, wenn der Trigger feuert, müssen sich in einer automatischen Aktivität hinter dem Trigger befinden! Damit wird eine gute Prozesslogik sichergestellt und Sie können den Prozess im Process Designer leichter visualisieren.

C.7.3.4 Geschäftslogik von Event-Triggern

Reihenfolge des Feuerns bei serialisierten Event-Triggern

Wenn ein für den Event-Trigger relevantes Ereignis eingetreten ist, feuert dieser Trigger. Danach wird das *Skript nach Event* ausgeführt. Wenn es *true* zurückgibt, wird die folgende automatische Aktivität oder der Entscheidungsknoten mit den beiden folgenden automatischen Aktivitäten ausgeführt.

Wenn der Bearbeiter mehr als einen Ticketparameter ändert und für diese Parameter im Bereich unterschiedliche Event-Trigger definiert wurden, feuern die Event-Trigger entsprechend ihrer Reihenfolge im Bereich.

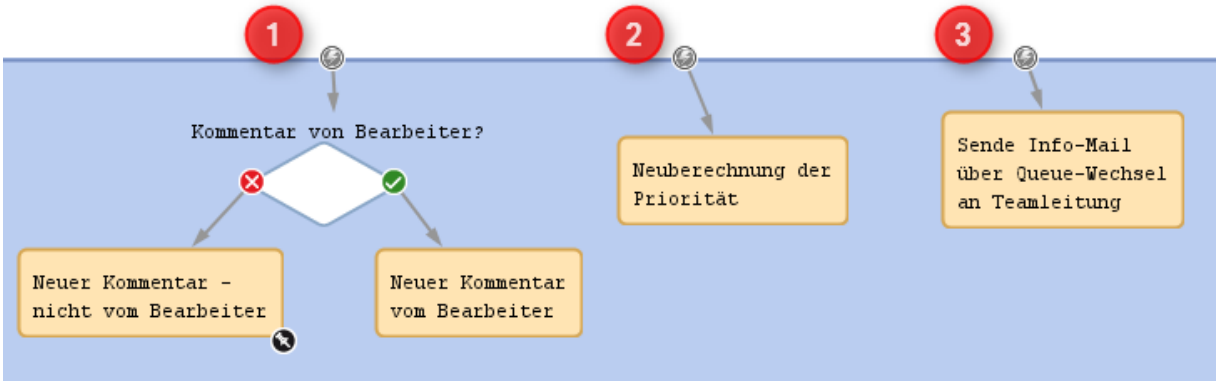


Abbildung 85: *ConSol CM Process Designer - Reihenfolge des Feuerns von Event-Triggern (1)*

Wenn einer der Event-Trigger das Ticket an eine neue Position verschiebt (d. h. es befindet sich danach nicht mehr in dem Bereich, in dem sich der nächste Event-Trigger befinden würde), feuert der folgende Event-Trigger nicht. Im Beispiel in der nächsten Abbildung feuert der Event-Trigger **(3)** nicht, wenn der Trigger *Neuberechnung der Priorität* gefeuert hat **(2)** (siehe *Anwendungsfall 2* im Abschnitt Beispiele für Event-Trigger), da die nachfolgende Aktion das Ticket in eine andere Queue übergeben hat.

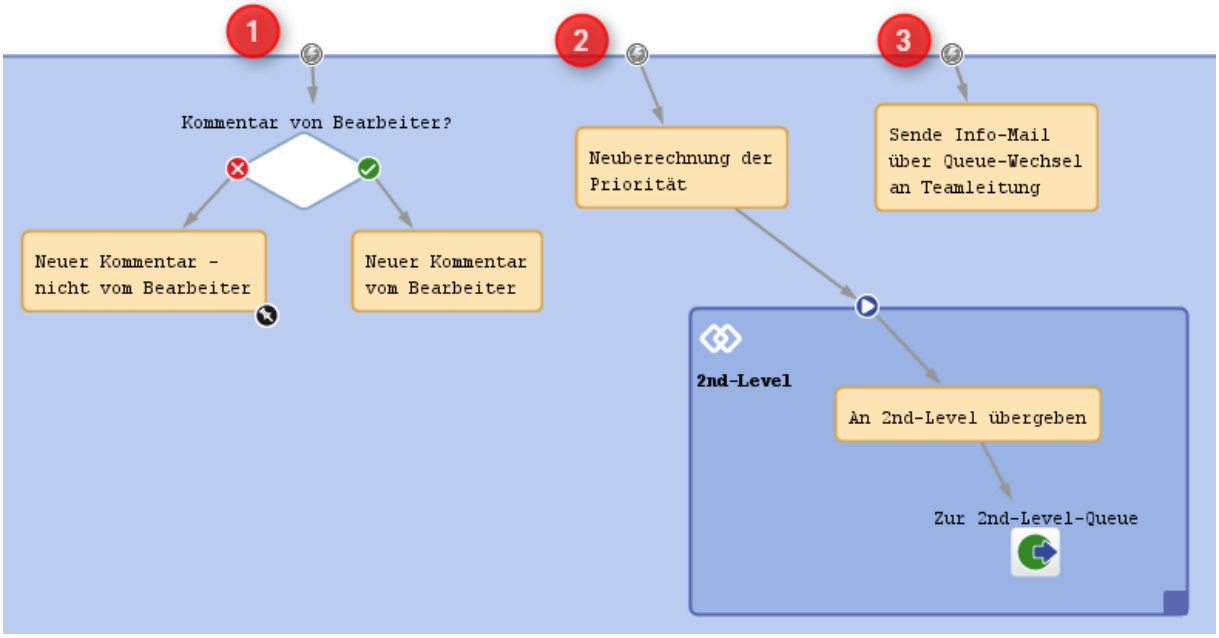


Abbildung 86: *ConSol CM Process Designer - Reihenfolge des Feuerns von Event-Triggern (2)*

Reihenfolge des Feuerns von Event-Triggern in hierarchischen Bereichen

Wenn sich Event-Trigger in hierarchischen Bereichen befinden, wird das Ereignis vom innersten Event-Trigger *verbraucht*, d. h. vom Event-Trigger im am weitesten innen liegenden Bereich. Alle Ereignisse, die nicht verbraucht wurden, werden vom nächst äußeren Bereich verarbeitet und so weiter.

Fall 1

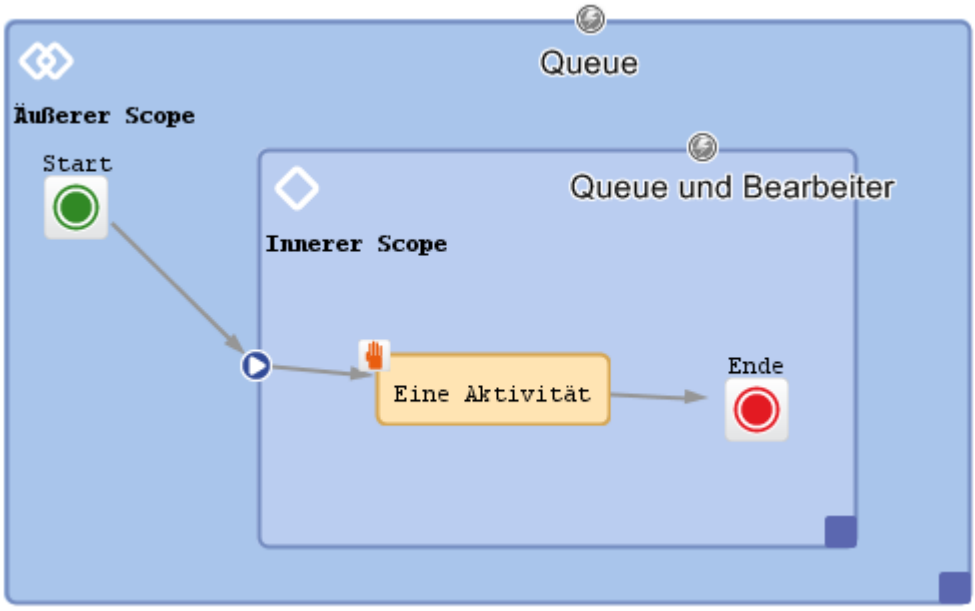


Abbildung 87: *ConSol CM Process Designer - Hierarchische Event-Trigger (1)*

Gefeuerte Ereignisse:

Ereignisse	Gefeuerte Trigger
Queue	Innerer
Queue und Bearbeiter	Innerer für beide
Bearbeiter	Innerer

Fall 2

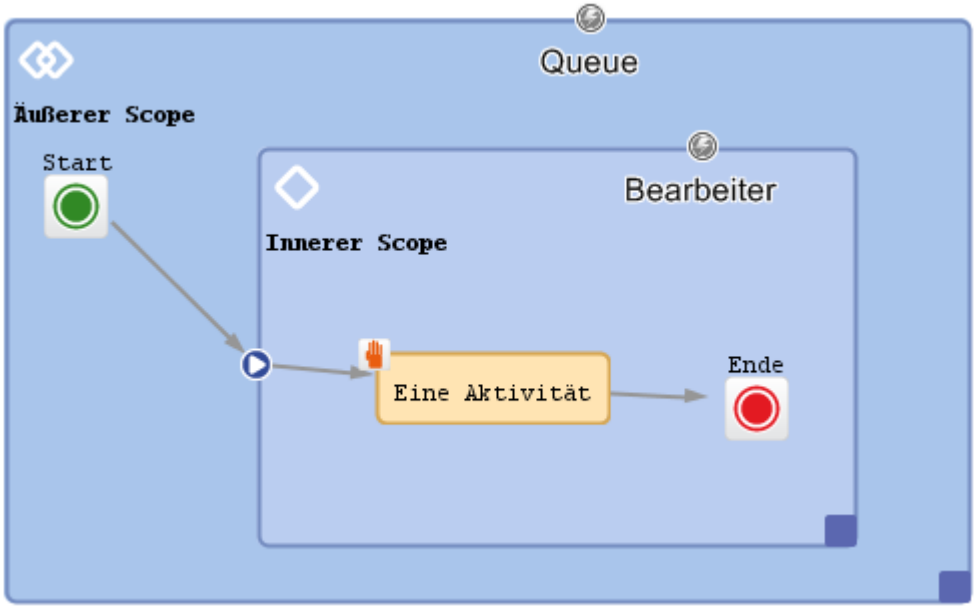


Abbildung 88: *ConSol CM Process Designer - Hierarchische Event-Trigger (2)*

Gefeuerte Ereignisse:

Ereignisse	Gefeuerte Trigger
Queue	Äußerer
Bearbeiter	Innerer
Queue und Bearbeiter	Innerer und äußerer

Fall 3

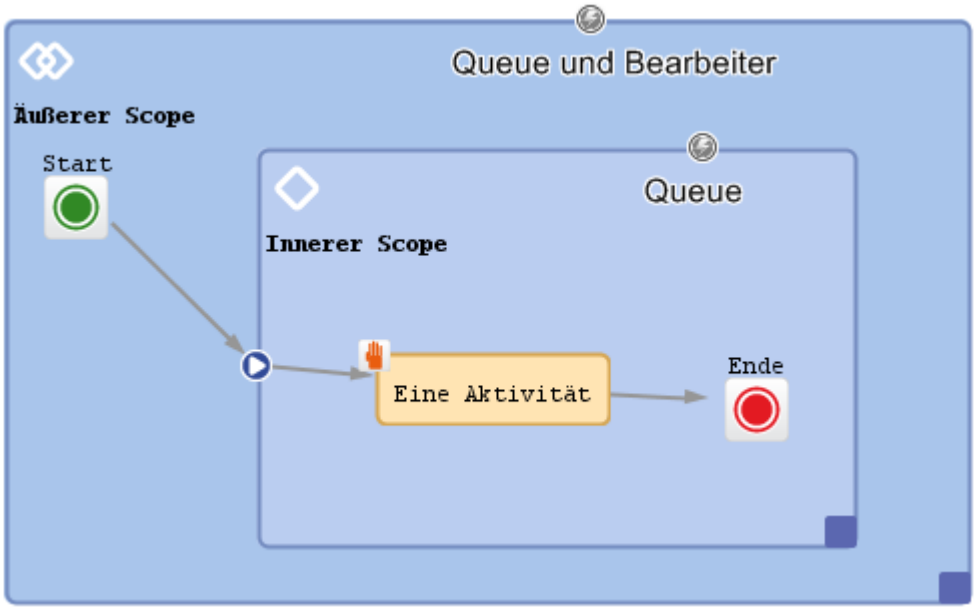


Abbildung 89: *ConSol CM Process Designer - Hierarchische Event-Trigger (3)*

Gefeuerte Ereignisse:

Ereignisse	Gefeuerte Trigger
Queue	Innerer
Bearbeiter	Äußerer
Queue und Bearbeiter	Innerer (Queue) und Äußerer (nur Bearbeiter)

C.7.3.5 Beispiele für Event-Trigger

Anwendungsfall 1: Kommentar des Bearbeiters prüfen

Wenn ein neuer Kommentar von jemanden hinzugefügt wurde, der nicht der aktuelle (zugewiesene) Bearbeiter des Tickets ist, wird ein Overlay zum Ticket-Icon hinzugefügt. Auf diese Weise wird das Ticket markiert und der Bearbeiter kann in der Ticketliste sehen, dass es einen neuen Kommentar in einem seiner Tickets gibt. Der Kommentar kann von einem anderen Bearbeiter gemacht worden sein, der Schreibrechte auf die Queue hat, oder von einem Kunden, der über ConSol CM.Track Kommentare hinzufügen kann. Oder es ist eine E-Mail eingegangen.

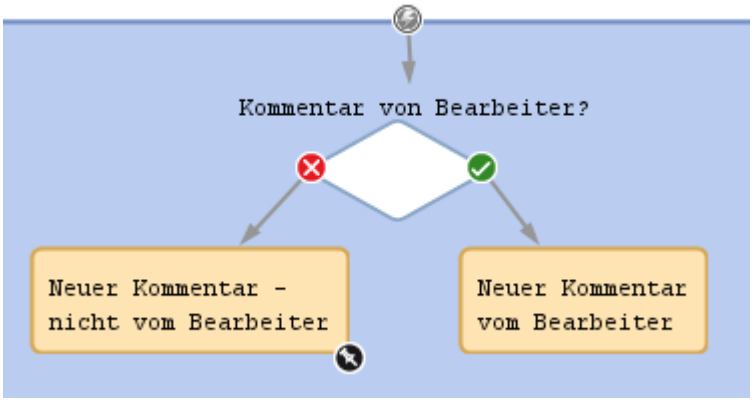


Abbildung 90: *ConSol CM Process Designer - Event-Trigger mit folgenden Aktivitäten*

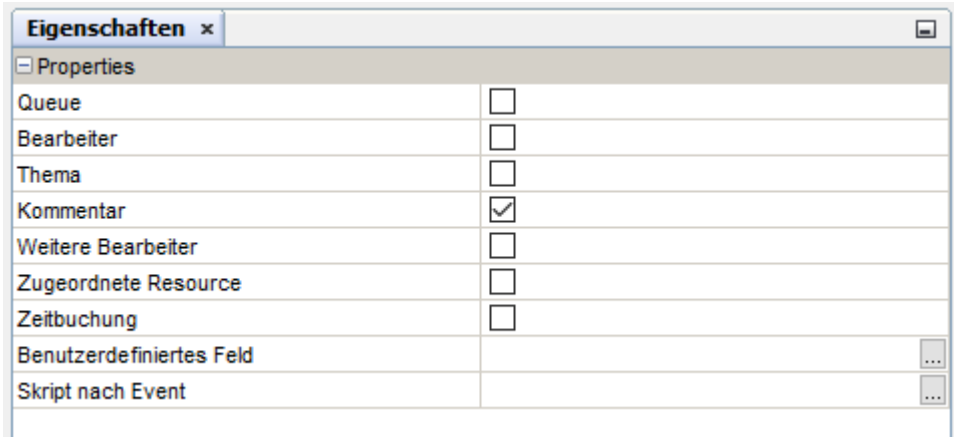


Abbildung 91: *ConSol CM Process Designer - Eigenschaften eines Event-Triggers (1)*

```
return (workflowApi.getCurrentEngineer() == ticket.getEngineer())
```

Code-Beispiel 9: *Code des Skripts des Entscheidungsknotens*

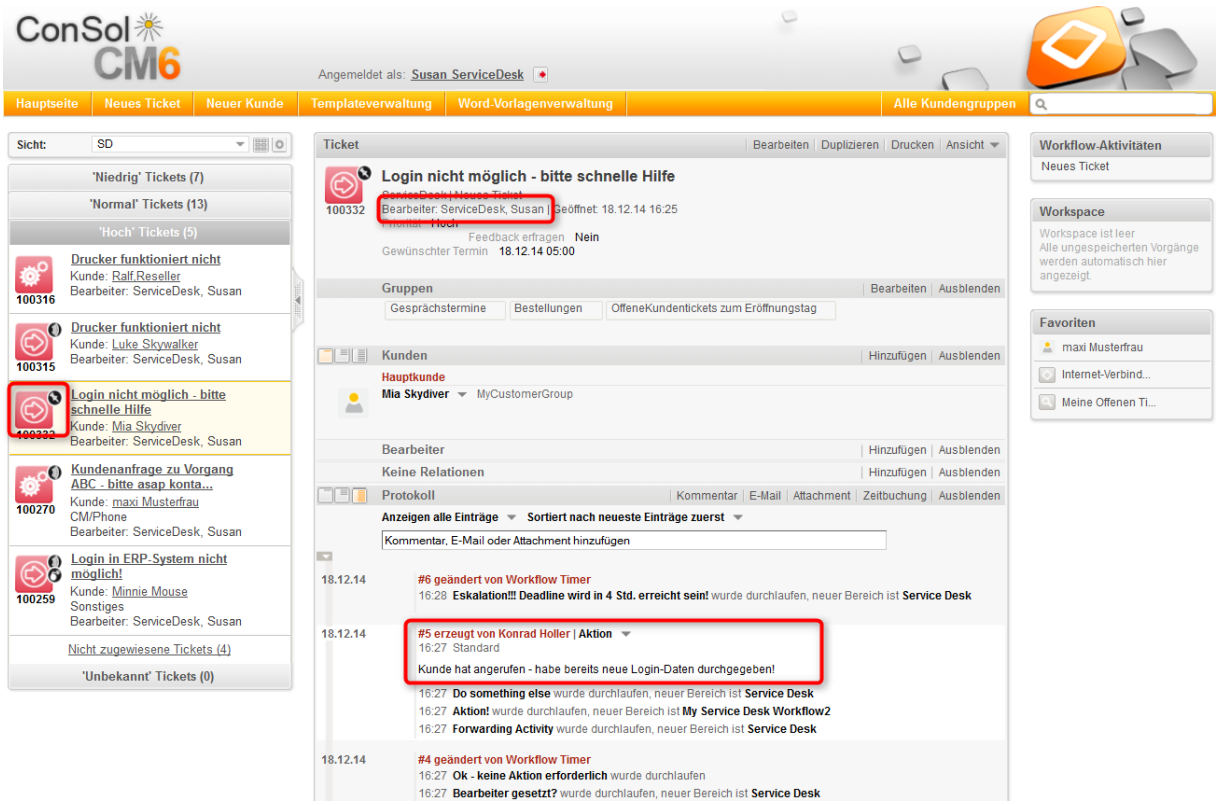


Abbildung 92: *ConSol CM Web Client- Mit neuem Overlay gekennzeichnetes Ticket*

Anwendungsfall 2: Neuberechnung der Ticketpriorität nach Änderung von Auswirkung und/oder Dringlichkeit

Dieses Beispiel stammt aus einer *ITIL-Servicedesk-Umgebung*. Nach den *ITIL-Standards* wird die Ticketpriorität anhand von zwei Werten berechnet: Auswirkung und Dringlichkeit. Das bedeutet, dass es im Ticket zwei Felder gibt, die vom Bearbeiter geändert werden können, und die Priorität automatisch aus diesen beiden Werten berechnet wird. Die Priorität kann das als Ticketfarbe oder als schreibgeschützte Liste (oder beides) angezeigt werden.

Dieses Prinzip erfordert eine Neuberechnung der Priorität, wenn mindestens eines der beiden Felder (Auswirkung/Dringlichkeit) geändert wurde. Dazu wird ein Event-Trigger mit einer nachfolgenden Aktivität, in der die Neuberechnung durchgeführt wird, erstellt.

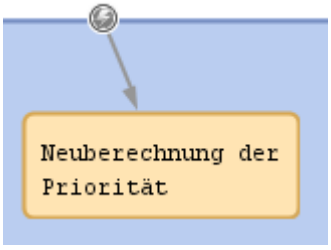


Abbildung 93: *ConSol CM Process Designer - Event-Trigger mit nachfolgender automatischer Aktivität*

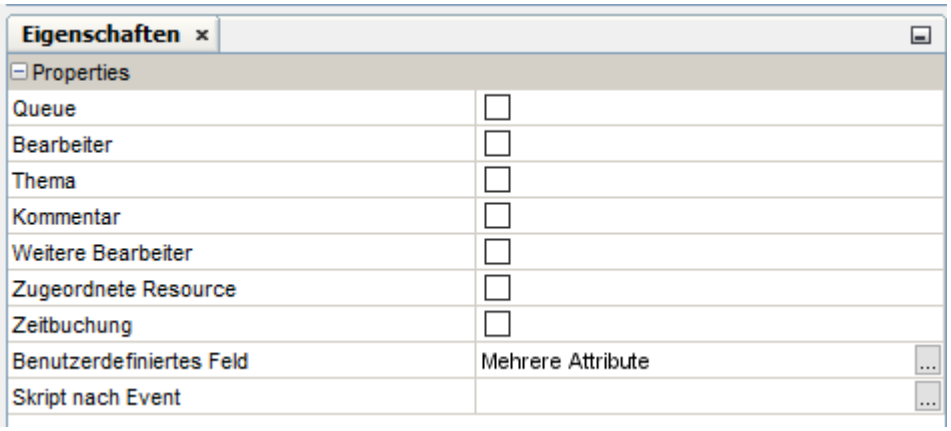


Abbildung 94: *ConSol CM Process Designer - Eigenschaften eines Event-Triggers (2)*

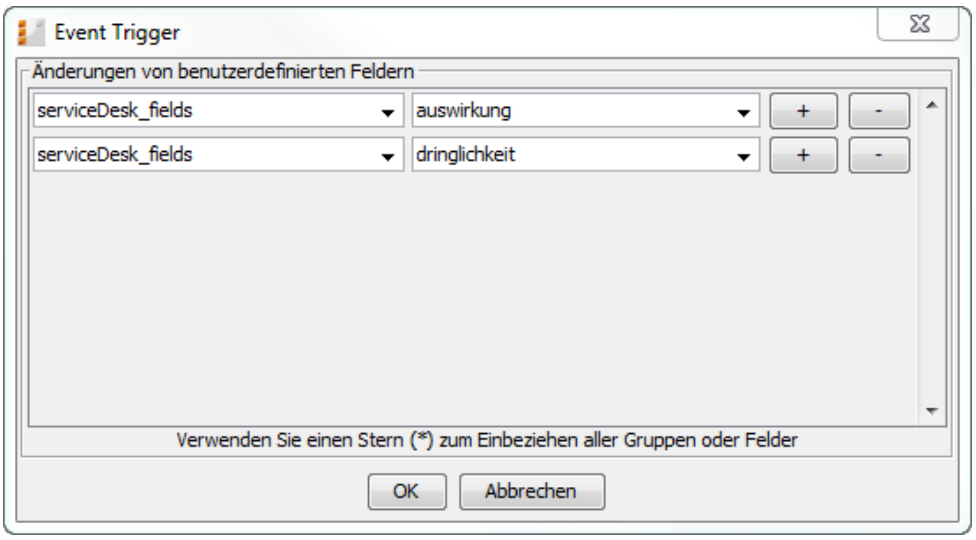


Abbildung 95: *ConSol CM Process Designer - Eigenschaft "Benutzerdefiniertes Feld" eines Event-Triggers (2)*

```
// Re-calculate priority:
String imp_value = ticket.get("service_desk_fields.impact").getName()
String urg_value = ticket.get("service_desk_fields.urgency").getName();

ScriptProvider scriptProvider = scriptProviderService.createDatabaseProvider
 ("calculatePriority.groovy")
//content of calculatePriority.groovy is omitted here, because it is not relevant
 for the current context
```

Code-Beispiel 10: *Code der automatischen Aktivität zur Neuberechnung der Priorität*

Anwendungsfall 3: Fortsetzen des Auslieferungsprozesses, wenn die Bestellung eingetroffen ist

Dies ist ein Beispiel aus einem Versand- und Auslieferungsprozess. Es werden neue Komponenten (z. B. Hardware) bestellt. Das Ticket wartet im Bereich *Bestellung: Auf Lieferung warten*. Wenn die Lieferung eingetroffen ist, erfasst ein anderes Team die Lieferung und setzt den Tag *Ware eingegangen*. Diese Änderung der Ticketdaten (*Ware eingegangen* von *false* in *true*) wird von einem Event-Trigger erfasst, der den Wert des entsprechenden *Boolean-Felds* (der Checkbox) überwacht. Nachdem der Event-Trigger gefeuert hat, wird die Checkbox (im Entscheidungsknoten) überprüft, und wenn der Wert auf *true* gesetzt ist, wird das Ticket in den nächsten Bereich *Lieferung Komponenten* geschoben. Die Bearbeiter, die für die Lieferung zuständig sind, sehen das Ticket jetzt in ihrer Sicht *Für Auslieferung bereite Komponenten* und können die Lieferung über *Alle Komponenten geliefert* bestätigen, wenn sie fertig sind.

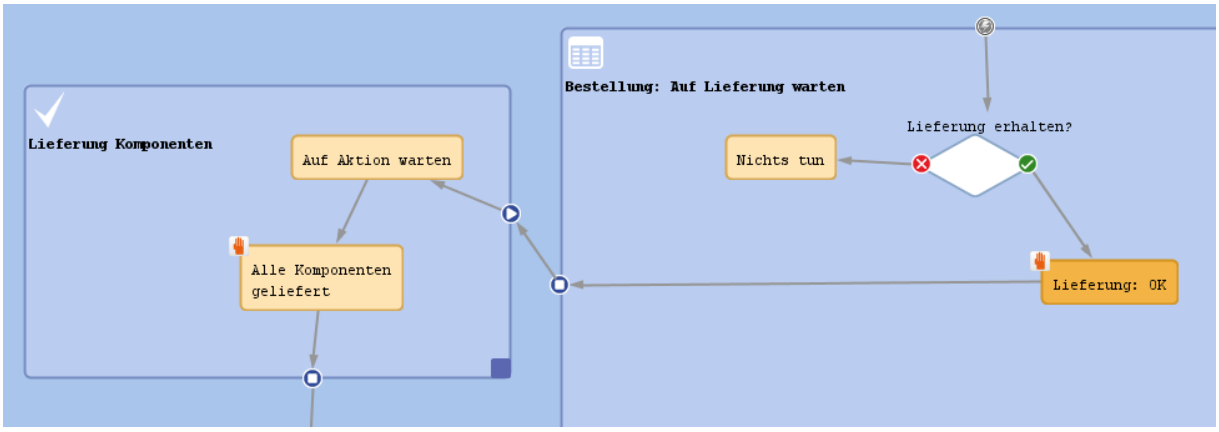


Abbildung 96: *ConSol CM Process Designer - Workflow für Anwendungsfall 3*

Anwendungsfall 4: Änderungen im Ticket für Reports erfassen

Wenn ein Event-Trigger feuert, dann hat sich im Ticket etwas geändert. Was sich geändert hat, können Sie in einem Report festhalten. Das Objekt, das die benötigten Informationen enthält, ist ein Objekt der Klasse *TicketChanges*.

Das folgende Beispiel zeigt, wie Sie die Änderung der Deadline in einem Aufgabenticket für das Reporting erfassen können. Wenn die Deadline sich ändert und das Ticket aktuell einem Bearbeiter zugewiesen ist, soll dieser Bearbeiter per E-Mail darüber informiert werden, dass sich das Datum geändert hat.

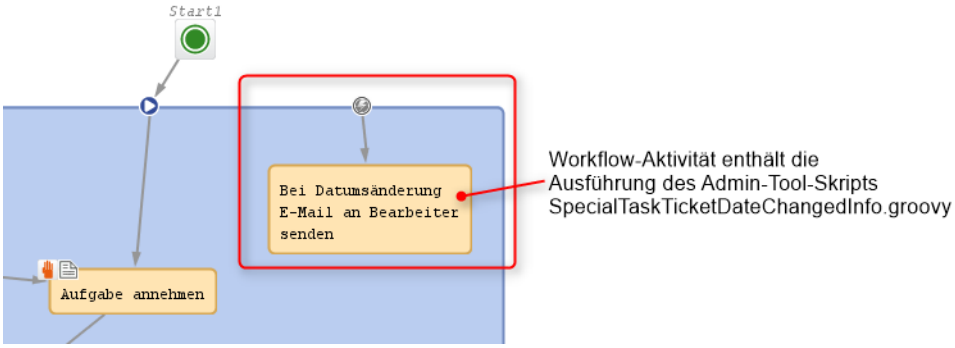


Abbildung 97: *Event-Trigger mit verknüpfter automatischer Aktivität. Im Beispiel überwacht der Trigger, ob sich das Benutzerdefinierte Feld "Deadline" im Ticket ändert (nicht sichtbar).*

```
import com.consol.cmas.common.model.customfield.meta.FieldKey
import com.consol.cmas.common.model.event.ticket.support.TicketChanges
import com.consol.cmas.common.model.mail.Mail
import com.consol.cmas.core.server.service.*
import static com.consol.cmas.common.util.TemplateUtil.TICKET_SUBJECT_TEMPLATE_NAME
log.info 'SpecialTaskTicketChangedInfo started ...'
// grep ticket changes
TicketChanges changes = workflowApi.getTicketUpdateEvent().getModifications()
def ticket = workflowApi.getTicket()
def eng = ticket.engineer
def new_date
def old_date
if (changes) {
   log.info 'Special task ticket ' + ticket.id + ' has been changed.'
   // if deadline field has been modified, send mail to current engineer if present
   if (eng) {
      // field key for requested field: deadline
      fk_deadl = new FieldKey("SpecialTasks_Fields","Deadline")
      mod_deadl = changes.getCustomFieldChangeInfo(fk_deadl)
      if (mod_deadl){
           // send email to engineer
           def mail = new Mail()
           mail.setTargetEngineer(eng)
           def replyaddress = configurationService.getValue("cmweb-server-
            adapter","mail.reply.to")
           mail.setReplyTo(replyaddress)
           def text = workflowApi.renderTemplate
            ("SpecialTasksTicketDeadlineChangedInfo")
           def text2 = 'The old date was ' + mod_deadl.previousValue.value + ' -- '
           def text3 = ' The new date is ' + mod_deadl.value.value
           mail.setText(text + text2 + text3)
           // def ticketName = ticket.getName()
           def subject = templateService.merge(TICKET_SUBJECT_TEMPLATE_NAME,
            [ticketName:ticket.name])
           def subject2 = " Deadline changed in Special Tasks ticket!"
           mail.setSubject(subject + subject2)

           try {
              mail.send();
           } catch (Exception e){
              mailStatus = false;
           }

      } // end if (mod_deadl)
   } // end if (eng)
// end if (changes)
```

Code-Beispiel 11: *Senden einer E-Mail an den aktuellen Bearbeiter des Tickets, wenn sich das Feld "Deadline" im Ticket geändert hat (Daten nicht formatiert)*

C.7.3.6 Best Practices: Verwendung von Event-Triggern

Siehe Abschnitt Vermeiden von sich selbst auslösenden Event-Triggern.

C.7.4 Aktivitäts-Formulare (ACFs)

In diesem Kapitel werden folgende Themen behandelt:

- Einführung in ACFs
- Hinzufügen eines ACFs zu einem Workflow
- Eigenschaften eines ACFs
- Geschäftslogik von ACFs
- Beispiele für die Verwendung von ACFs

C.7.4.1 Einführung in ACFs

Ein *Aktivitäts-Formular (Activity Control Form = ACF)* ist ein Webformular, das dem Bearbeiter in einem oder mehreren Prozessschritten angezeigt wird. Auf diese Weise kann die Dateneingabe sehr genau kontrolliert werden.

Abbildung 98: *ConSol CM Process Designer - Aktivitätsformular (ACF)*

Zum Beispiel kann ein Helpdesk-Mitarbeiter eine Beschwerde nicht ohne Angabe von Gründen ablehnen. Dies ist im Prozess über ein ACF implementiert, das angezeigt wird, wenn der Bearbeiter auf die Workflow-Aktivität *Ticket verwerfen* klickt. Es wird ein Formular geöffnet, in dem der Bearbeiter eine Kategorie für die Ablehnung auswählen muss und eine Bemerkung in das Textfeld eingeben kann. Oder, am Beispiel eines Sales-Prozesses, wenn ein Bearbeiter (in diesem Fall ein Vertriebsmitarbeiter) auf *Termin mit potentiellem Kunden vereinbaren* klickt, wird ein Formular angezeigt, in dem er das Budget, die Größe der Firma des Kunden und die Produkte, für die sich der Kunde interessiert, eingeben kann.

Ein ACF kann optionale Felder und Pflichtfelder enthalten.

> Wir empfehlen, hinter den Namen aller Aktivitäten, bei denen automatisch ein ACF geöffnet wird, "..." zu schreiben. So können die Bearbeiter einfacher zwischen normalen Aktivitäten und Aktivitäten mit ACF unterscheiden.

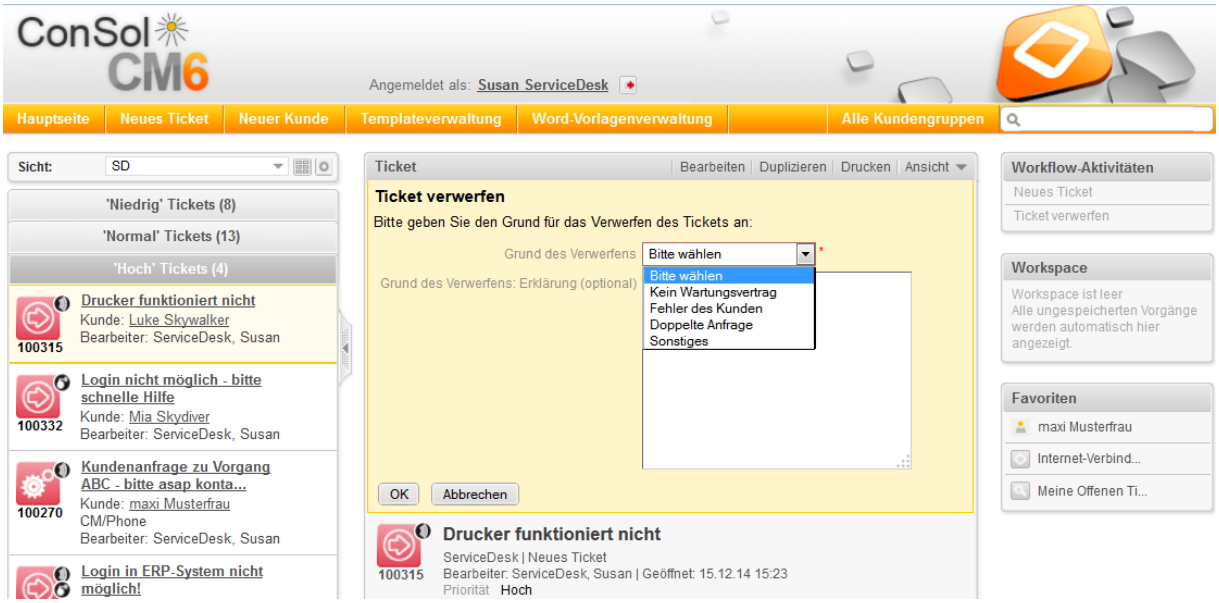


Abbildung 99: *ConSol CM Web Client - Geöffnetes ACF*

C.7.4.2 Hinzufügen eines ACFs zu einem Workflow

Variante A: Starten der Definition des ACFs mit dem Admin Tool

Bevor Sie ein ACF im Workflow hinzufügen können, müssen Sie es mit dem Admin Tool definieren. Eine detaillierte Erklärung dazu finden Sie im *ConSol CM Administratorhandbuch*, Kapitel *Verwaltung von Benutzerdefinierten Feldern*. Im vorliegenden Handbuch gehen wir davon aus, dass Sie das ACF bereits definiert haben und es zum Workflow hinzufügen möchten.

Ein ACF wird immer zu einer manuellen Aktivität hinzugefügt. Um ein ACF zur Zielaktivität hinzuzufügen, klicken Sie in der Palette auf das ACF-Symbol und ziehen Sie es in die gewünschte Aktivität. Dann können Sie die Eigenschaften des ACF konfigurieren. Wenn Sie ein ACF zu einer automatischen Aktivität hinzufügen, wird der Typ dieser Aktivität in *Manuell* geändert.

Im Web Client wird das ACF geöffnet, wenn der Benutzer auf die Workflow-Aktivität klickt, an die das ACF im Workflow angehängt ist. Siehe obige Abbildung.

Variante B: Starten der ACF-Definition mit dem Process Designer

Sie können auch ein leeres ACF zu einer Workflow-Aktivität hinzufügen und den Namen dabei definieren. Dann wird ein leeres ACF im Admin Tool erstellt und Sie müssen die Benutzerdefinierten Felder dem ACF in einem späteren Schritt zuweisen.

> ⚠ Vergessen Sie nicht, die Daten im Admin Tool neu zu laden! Wenn Sie ein ACF im Process Designer definiert haben, findet keine automatische Datenübertragung an das Admin Tool statt.

C.7.4.3 Eigenschaften eines ACFs

Ein ACF hat folgende Eigenschaften:

- **Name**
  String. Der Name des ACFs. Wählen Sie den Namen aus dem Drop-down-Menü. Es sind alle im Admin Tool definierten ACFs verfügbar.
- **Pflichtfelder**
  Damit wird ein Pop-up-Fenster geöffnet (siehe obige Abbildung), in dem Sie die Pflichtfelder definieren können. Standardmäßig sind alle ACF-Felder optional, d. h. wenn das Formular im Web Client geöffnet wird, kann der Bearbeiter Daten eingeben, den Prozess aber auch ohne Dateneingabe fortsetzen. Bei Pflichtfeldern kann der Prozess nur fortgesetzt werden, wenn die Pflichtfelder ausgefüllt sind.
- **Initialisierungsskript**
  Hier können Sie ein Skript definieren, das ausgeführt wird, bevor das ACF geladen wird. Normalerweise wird so ein Skript verwendet, um Standardwerte für die Benutzerdefinierten Felder des ACFs zu setzen.
- **Vorbedingungsskript**
  Hier können Sie ein Skript definieren, das ausgeführt wird, um festzulegen, ob das ACF angezeigt wird (Rückgabewert *true*) oder nicht (Rückgabewert *false*).

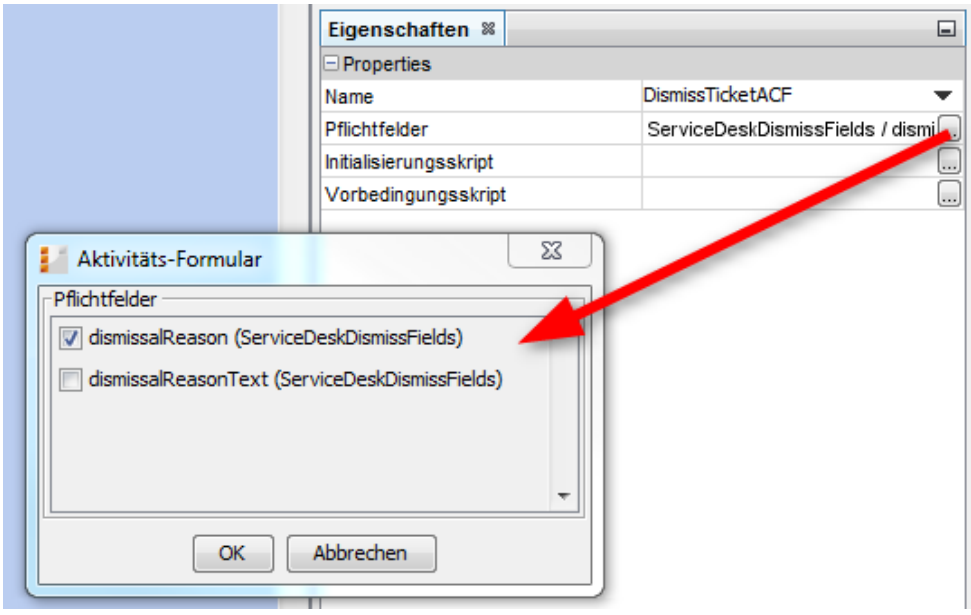


Abbildung 100: *ConSol CM Process Designer - Eigenschaften eines ACFs*

⚠ Alle Benutzerdefinierten Felder, die zum ACF gehören, müssen in der Ziel-Queue verfügbar sein, d. h. die entsprechende Benutzerdefinierte Feldgruppe muss der Queue zugewiesen sein, in der der Workflow verwendet wird! Es gibt zwei Möglichkeiten, um dies zu erreichen:

1. Sie weisen die Benutzerdefinierte Feldgruppe der Queue manuell zu.
2. Sie erstellen das ACF und verwenden es in einem Workflow. Wenn Sie den Workflow installieren, weist ConSol CM die erforderlichen Benutzerdefinierten Feldgruppen automatisch den Queues zu, in denen der Workflow verwendet wird.

Eine detaillierte Erklärung der Queue-Verwaltung finden Sie im *ConSol CM Administratorhandbuch*.

C.7.4.4 Geschäftslogik von ACFs

ACF in einer manuellen Aktivität

ACF sind nur für manuelle Aktivitäten möglich. Wenn ein Benutzer im Web Client eine Workflow-Aktivität auswählt, die ein ACF hat, werden folgende Schritte ausgeführt:

1. Wenn es ein **ACF-Vorbedingungsskript** gibt, wird das Vorbedingungsskript ausgeführt.

   Wenn das ACF-Vorbedingungsskript *true* zurückgibt:

   a. Wenn es ein **ACF-Initialisierungsskript** gibt: Das ACF-Initialisierungsskript wird ausgeführt.

   b. Das **ACF** wird angezeigt und der Bearbeiter füllt das Formular aus, mit optionalen Feldern und Pflichtfeldern. Wenn Felder, die zum ACF gehören, auch als normale Ticketdatenfelder verfügbar sind, kann es sein, dass diese Felder schon vor Öffnen des ACFs von einem Bearbeiter editiert/ausgefüllt wurden. Diese Felder können daher im ACF bereits ausgefüllt sein. Der Bearbeiter kann sie so belassen (und das ACF nur zur Kontrolle verwenden) oder den Inhalt der Felder ändern.

   c. Die **Workflow-Aktivität** wird ausgeführt, sobald der Bearbeiter im ACF auf *OK* geklickt hat.

   Wenn das ACF-Vorbedingungsskript *false* zurückgibt:

   a. Das **ACF** wird nicht angezeigt.

   b. Die **Workflow-Aktivität** wird ausgeführt, sobald der Bearbeiter im Web Client auf die Workflow-Aktivität (den Namen) geklickt hat.

Wenn ein ACF **abgebrochen** wird, kehrt das Ticket zum Bereich der letzten Aktivität zurück, da das Ticket immer **hinter** der letzten Aktivität wartet (und **nicht** vor der nächsten).

Wenn die Daten des ACFs erst angezeigt werden sollen, wenn ein bestimmter Schritt im Prozess erreicht wurde, können die Daten in eine (oder mehrere) eigene Benutzerdefinierte Feldgruppen geschrieben werden, die am Anfang des Prozesses *unsichtbar* sind (Annotation *group-visibility = false* für die Benutzerdefinierte Feldgruppe). Im Schritt nach der Aktivität mit dem ACF wird die Benutzerdefinierte Feldgruppe mithilfe des Skripts einer Workflow-Aktivität eingeblendet. Empfehlungen zur Verwendung von ACFs finden Sie im Abschnitt Best Practices dieses Handbuchs.

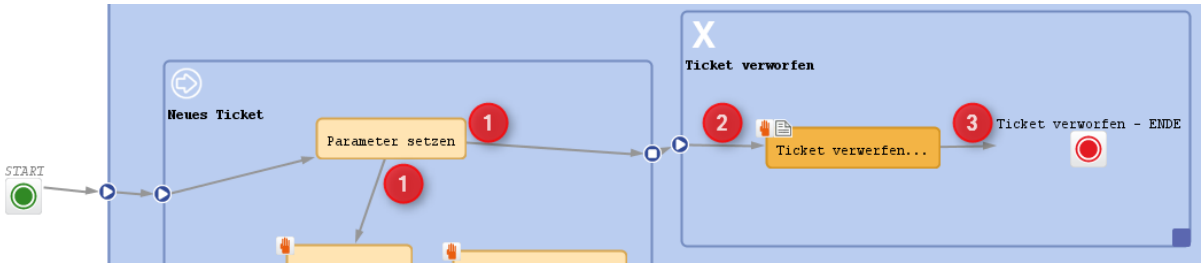


Abbildung 101: *ConSol CM Process Designer - ACF-Prozesslogik*

Beispiel (ACF mit Initialisierungsskript, ohne Vorbedingungsskript):

- Ein Ticket wird erstellt und durchläuft die automatische Aktivität *Parameter setzen*.
- Es wartet hinter dieser Aktivität an der Position **(1)** im Bereich *Neues Ticket*. Die nächsten Aktivitäten *Ticket verwerfen ...* und *Neues IT-Ticket* (hier nicht abgebildet) werden im Web Client angezeigt.
- Der Bearbeiter wählt *Ticket verwerfen ... .*
- Das Initialisierungsskript für das ACF in *Ticket verwerfen ...* wird ausgeführt **(2)**.
- Das ACF wird auf der GUI angezeigt.
    - **Variante 1:**
        1. Das ACF wird abgebrochen.
        2. Das Ticket kehrt zu **(1)** zurück.
    - **Variante 2:**
        1. Das ACF wird ausgefüllt und bestätigt.
        2. Die Aktivität *Ticket verwerfen ...* wird ausgeführt (wenn diese Aktivität ein Skript hat, wird das Skript ausgeführt). Das Ticket durchläuft den Knoten und setzt seinen Weg fort **(3)**. Im obigen Beispiel wird es geschlossen.

ACF an manuellen Aktivität mit Bedingung

Wenn eine manuelle Aktivität eine Bedingung hat, wird die Aktivität nur angezeigt, wenn das Bedingungsskript *true* zurückgibt, d. h. das ACF wird nur angezeigt, wenn das Bedingungsskript *true* zurückgibt.

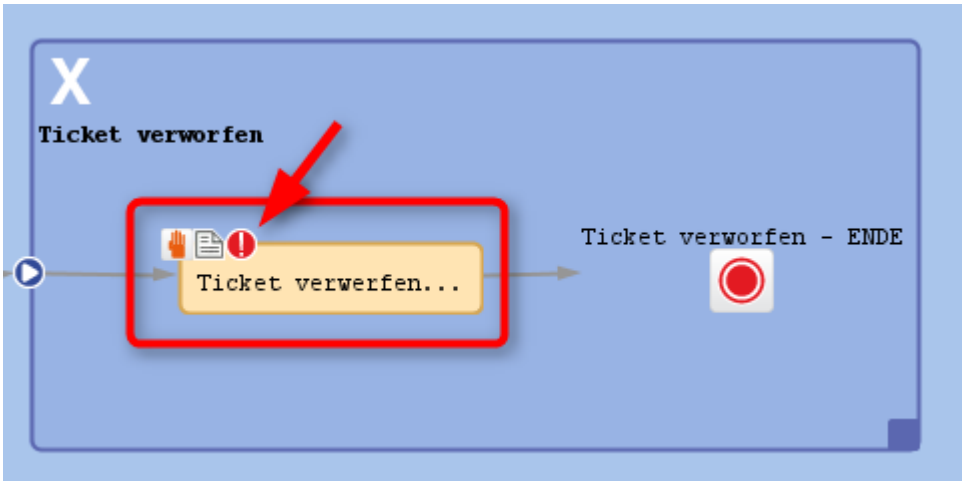


Abbildung 102: *ConSol CM Process Designer - Manuelle Aktivität mit ACF und Bedingung*

C.7.4.5 Beispiele für die Verwendung von ACFs

Anwendungsfall 1: ACF zum Ablehnen einer Kundenanfrage

Das Beispiel wurde in den vorherigen Abschnitten verwendet. Der Bearbeiter kann eine Kundenanfrage nur ablehnen, wenn er einen Grund angibt. Dieser wird aus der Drop-down-Liste ausgewählt. Außerdem kann der Bearbeiter eine Bemerkung in ein Textfeld eingeben.

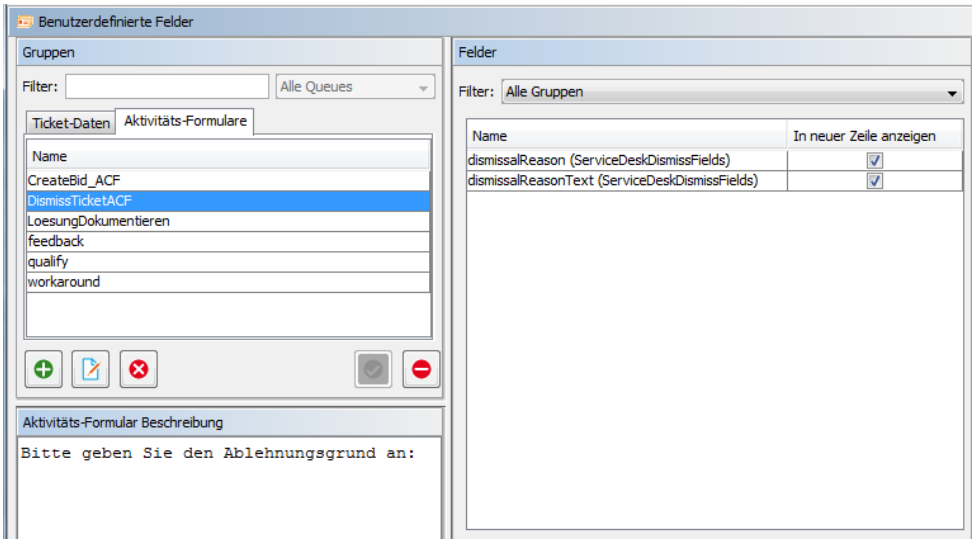


Abbildung 103: *ConSol CM Admin Tool - ACF-Definition*

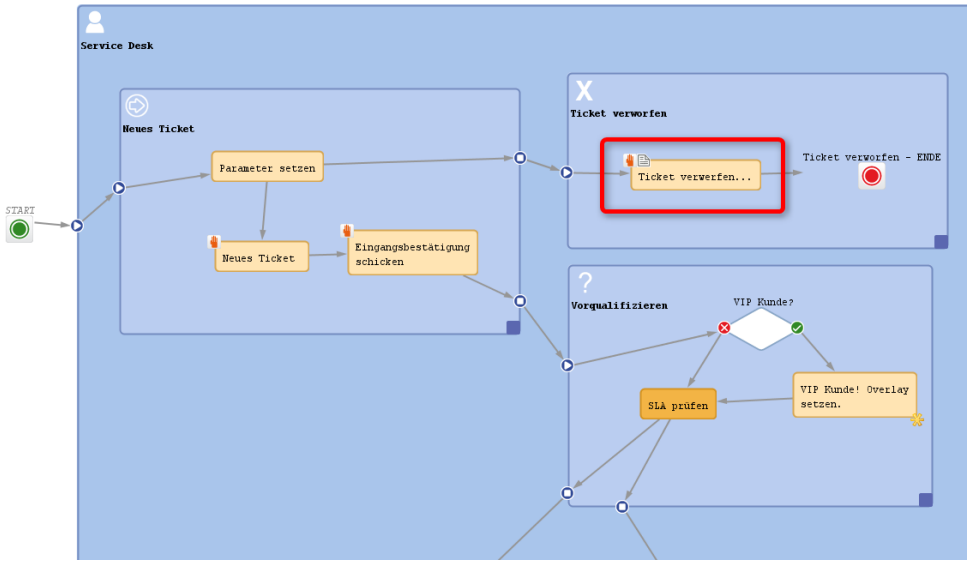


Abbildung 104: *ConSol CM Process Designer - ACF in Workflow*

Die GUI des Web Clients und die ACF-Eigenschaften sind in den Abbildungen in den vorherigen Absätzen gezeigt.

Anwendungsfall 2: Ausfüllen von Sales-Informationen beim Erstellen eines Angebots

Wenn ein Vertriebsmitarbeiter die Workflow-Aktivität *Angebot erstellen* auf der GUI des Web Clients auswählt, wird ein ACF geöffnet, das mehrere Felder enthält. Ein Feld ist ein Drop-down-Menü, bei dem per Skript ein Standardwert gesetzt wird. Die anderen Felder sind optional. Das Feld *Produkt* wurde für das Ticket in vorangegangenen Prozessschritten ausgefüllt, sodass es mit dem ausgewählten Wert angezeigt wird. Dieser kann entweder unverändert belassen werden oder geändert werden.

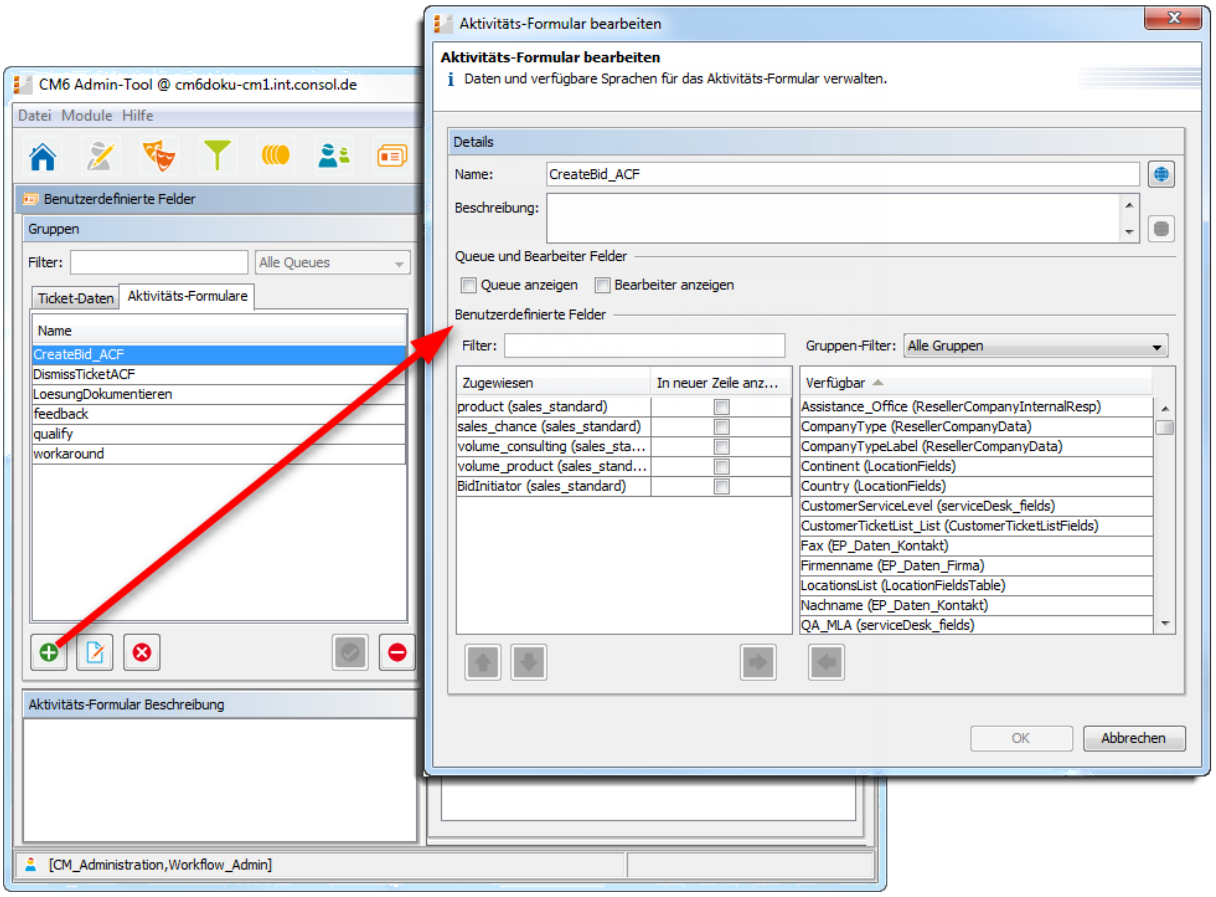


Abbildung 105: *ConSol CM Admin Tool - ACF für den Sales-Workflow*

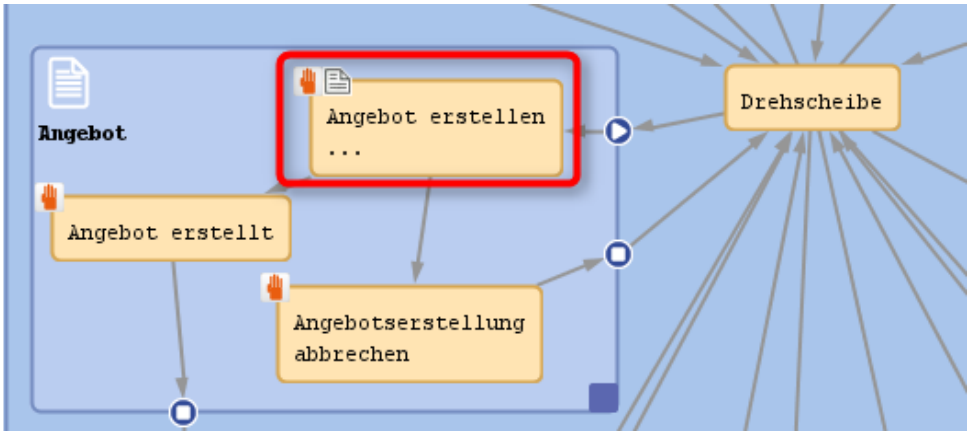


Abbildung 106: *ConSol CM Process Designer - ACF im Sales-Workflow*

```
ticket.set("sales_standard.BidInitiator","Mr. Miller")
```

Code-Beispiel 12: *Process Designer: Initialisierungsskript für das ACF zum Angebotserstellung*

Abbildung 107: *ConSol CM Web Client - ACF im Sales-Prozess*

C.8 Aussprung- und Einsprungknoten

In diesem Kapitel werden folgende Themen behandelt:

C.8.1 Einleitung

Ein Prozess besteht häufig aus einem oder mehreren Teilprozessen, z. B. kann es in einem IT-Helpdesk ein First-Level-Team geben, das Tickets akzeptiert und qualifiziert, und ein Second-Level-Team, das verschiedene Probleme lösen kann, und ein Third-Level-Team mit Spezialisten. Wenn Sie dies im Prozess abbilden möchten, müssen Sie für jeden Teilprozess (First-Level, Second-Level und Third-Level) einen Workflow erstellen. Die Teilprozesse müssen dann verknüpft werden, damit die Übergabe von Tickets von einem Team an das andere Team im Prozess auf die richtige Weise geschieht.

Ein Ticket kann vom First-Level-Team an das Second-Level-Team übergeben werden, dann geht es weiter zum Third-Level-Team und mit einer anderen Frage zurück an das Second-Level Team. Das Second-Level-Team übergibt es wieder an das Third-Level-Team und am Ende landet das Ticket wieder beim First-Level-Team, das den Kunden kontaktiert. Deshalb sind Verbindungen von einem Teilprozess zum nächsten erforderlich, d. h. ein Knoten, an dem ein Ticket den aktuellen Workflow verlässt (**Aussprungknoten**) und das Gegenstück im folgenden Workflow (**Einsprungknoten**). Wenn das Ticket am *Startknoten* des neuen Prozesses starten soll, ist kein Einsprungknoten erforderlich.

Im Process Designer werden Einsprungknoten und Aussprungknoten in den Workflow eingefügt, indem sie aus der Palette gezogen werden und entsprechend dem gewünschten Prozess mit anderen Workflow-Elementen verknüpft werden.

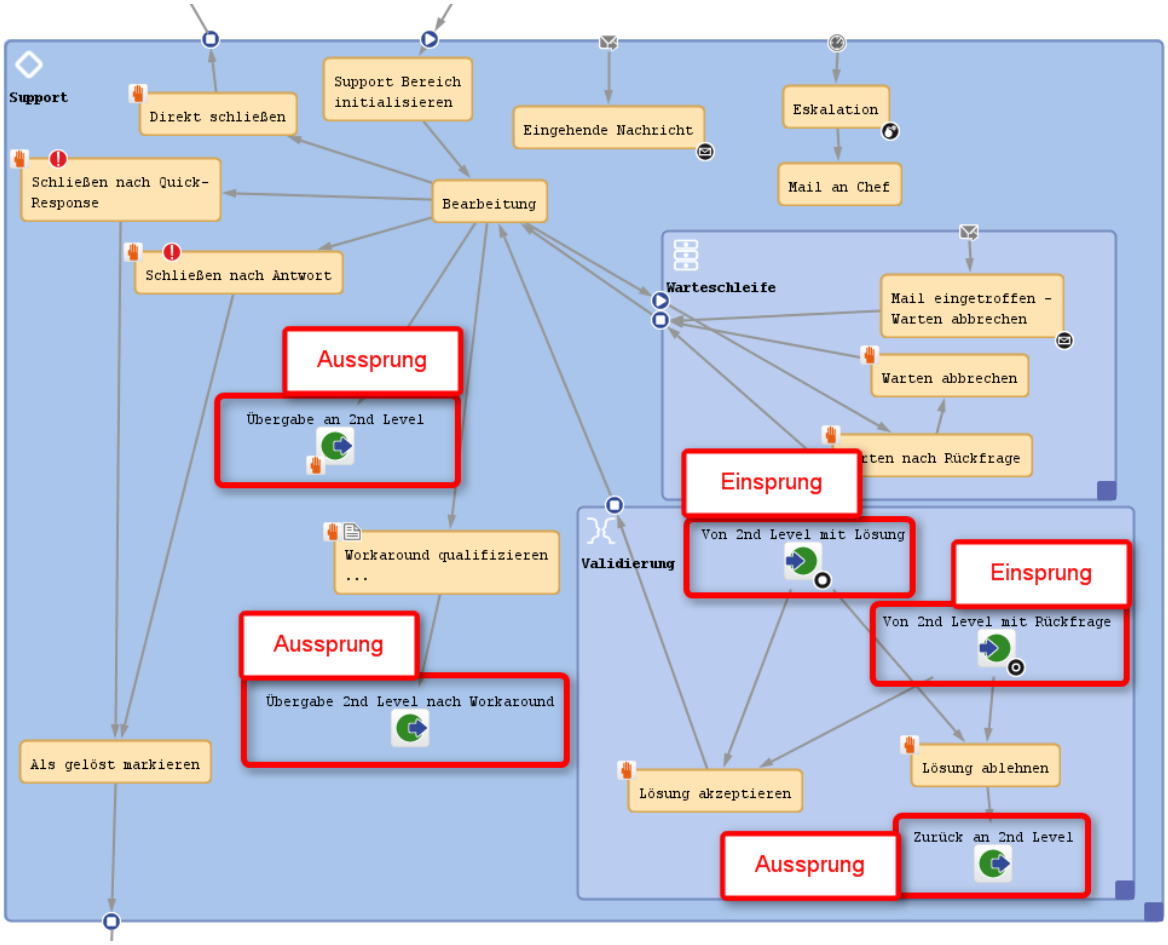


Abbildung 108: *ConSol CM Process Designer - Beispiel für Einsprung- und Aussprungknoten*

C.8.2 Aussprungknoten

Ein Aussprungknoten definiert eine Position im Prozess, an der das Ticket den Teilprozess verlassen und den nächsten Teilprozess beginnen soll.

Abbildung 109: *ConSol CM Process Designer - Aussprungknoten*

C.8.2.1 Eigenschaften eines Aussprungknotens

Für einen Aussprungknoten können folgende Eigenschaften definiert werden:

- **Name**
  String. Pflichtfeld. Technischer Name des Objekts
- **Bezeichnung**
  String. Optional. Lokalisierter Name (wenn nicht gesetzt, wird der technische Name verwendet), der auf der GUI des Web Clients angezeigt wird.
- **Beschreibung**
  String. Optional. Wird im Web Client als Mouseover angezeigt.
- **Sortier-Index**
  Auswahlfeld. Pflichtfeld. Definiert die Reihenfolge der Aktivitäten im Web Client.
- **Aussprungknotentyp**
  Auswahlfeld. Pflichtfeld. Es muss entweder *Automatisch* oder *Manuell* ausgewählt werden. Wenn es sich um einen manuellen Knoten handelt, ist der Knoten auf der GUI des Process Designers mit dem Icon *Hand* gekennzeichnet.
- **Skript**
  Optional. Es kann ein Skript definiert werden, das ausgeführt wird, wenn das Ticket in den Knoten wandert.
- **Name der Ziel-Queue**
  Auswahlfeld. Pflichtfeld. Wählen Sie den Namen der Queue, an die das Ticket übergeben werden soll.
- **Ziel-Einsprungknoten**
  Auswahlfeld. Optional. Wählen Sie den Einsprungknoten aus dem Drop-down-Menü. Es werden alle Einsprungknoten im Workflow der ausgewählten Queue angeboten. Wenn kein Einsprungknoten ausgewählt wird, beginnt das Ticket den anderen Prozess, d. h. die Ziel-Queue, am *Startknoten*.

> Wenn Sie mit der Workflow-Entwicklung beginnen, haben Sie möglicherweise ein *Henne-Ei-Problem*, wenn Sie mit der Definition von Aussprung- und Einsprungknoten beginnen, da Sie natürlich mit einem Workflow anfangen müssen, bevor es den anderen Workflow gibt. Wir empfehlen Ihnen, mit Dummy-Queues ohne spezielle Einsprungknoten zu arbeiten. Fügen Sie zu einem späteren Zeitpunkt die Namen der richtigen Ziel-Queue und die Einsprungknoten hinzu.

- **Ticket-Protokoll Sichtbarkeit**
  Auswahlfeld. Siehe Abschnitt Ticket-Protokoll Sichtbarkeit.
- **Autom. Aktualisierung deaktivieren**
  Boolean. Siehe Abschnitt Autom. Aktualisierung deaktivieren.

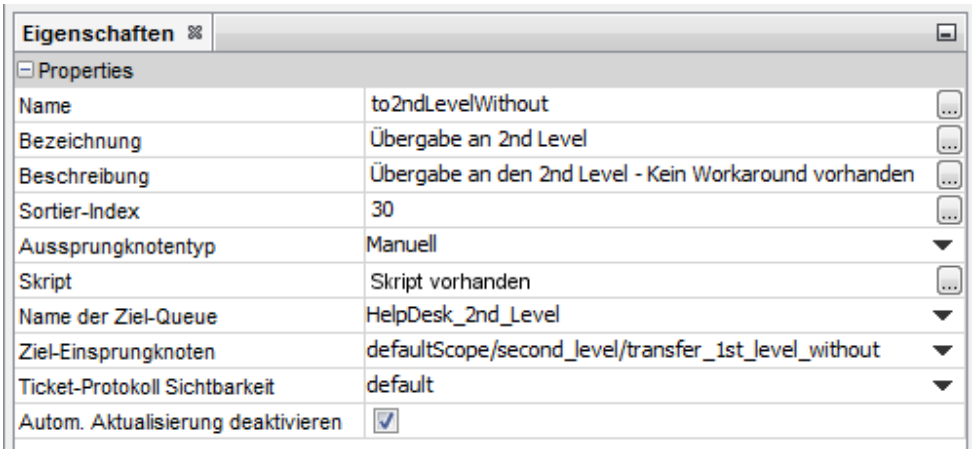


Abbildung 110: *ConSol CM Process Designer - Aussprungknoten: Eigenschaften-Editor*

C.8.3 Einsprungknoten

Ein Einsprungknoten ist der Knoten, der die Position definiert, an der ein Ticket aus einem anderen Prozess (Queue) in eine Queue mit dem aktuellen Workflow eintreten kann. Alle Einsprungknoten eines Workflows werden als Ziel-Einsprungknoten angeboten, wenn die Queue mit dem entsprechenden Workflow als Ziel-Queue für einen Aussprungknoten ausgewählt wird.

Abbildung 111: *ConSol CM Process Designer - Einsprungknoten*

C.8.3.1 Eigenschaften eines Einsprungknotens

Für einen Einsprungknoten können folgende Eigenschaften definiert werden:

- **Name**
  String. Pflichtfeld. Technischer Name des Objekts
- **Bezeichnung**
  String. Optional. Lokalisierter Name (wenn nicht gesetzt, wird der technische Name verwendet), der auf der GUI des Web Clients angezeigt wird.
- **Beschreibung**
  String. Optional. Wird im Web Client als Mouseover angezeigt.
- **Skript**
  Optional. Es kann ein Skript definiert werden, das ausgeführt wird, wenn das Ticket in den Knoten wandert.
- **Overlay**
  Auswahlfeld. Optional. Klicken Sie in das orangene Feld, um eines der Standard-Overlays von ConSol CM auszuwählen, oder verwenden Sie den Dateibrowser (...), um ein Icon aus dem Dateisystem hochzuladen.
- **Overlay-Gültigkeit**
  Auswahlfeld. Wird nur angezeigt, wenn ein Overlay ausgewählt wurde.
  - **Aktivität**
    Das Overlay hängt so lange am Ticket-Icon, wie das Ticket hinter der Aktivität steht. Sobald die nächste Aktivität ausgeführt wird, wird das Overlay vom Ticket-Icon entfernt.
  - **Bereich**
    Das Overlay wird gelöscht, wenn das Ticket den Bereich verlässt.
  - **Prozess**
    Das einmal zum Ticket-Icon hinzugefügte Overlay hängt für den Rest des Prozesses am Ticket-Icon.

- **Nächstes Overlay**
  Das Overlay hängt so lange am Ticket-Icon, wie kein neues Overlay hinzugefügt wird. Wenn ein neues Overlay hinzugefügt wird, wird das alte Overlay gelöscht.

- **Ticket-Protokoll Sichtbarkeit**
  Auswahlfeld. Siehe Abschnitt Ticket-Protokoll Sichtbarkeit.

- **Autom. Aktualisierung deaktivieren**
  Boolean. Siehe Abschnitt Autom. Aktualisierung deaktivieren.

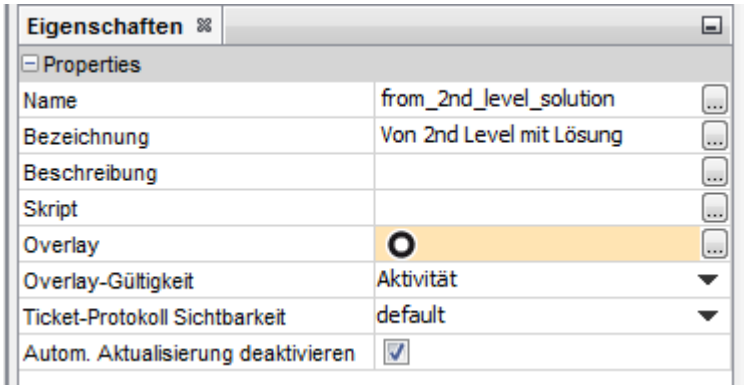


Abbildung 112: *ConSol CM Process Designer - Einsprungknoten: Eigenschaften-Editor*

D - Einführung in die Workflow-Programmierung

In diesem Kapitel werden folgende Themen behandelt:

D.1 Programmieren von CM-Skripten

Wie Sie in den vorherigen Abschnitten gesehen haben, können ConSol-Workflows über die **grafische Benutzeroberfläche des Process Designers** ziemlich einfach eingerichtet werden. Um "echte Intelligenz" in die Workflows zu bringen, müssen Sie allerdings programmieren, d. h. **ConSol CM-Workflow-Skripte** schreiben, die in den Workflow-Aktivitäten und Bedingungen verwendet werden.

ConSol CM-Skripte werden in Groovy geschrieben. Sie sollten also zumindest ein Grundwissen dieser Programmiersprache haben. Da der Groovy-Code auf einer Java Virtual Machine ausgeführt wird, können Sie auch Java-Code schreiben. Wenn Sie also Java- oder Groovy-Entwickler sind, können Sie einfach lernen, mithilfe der ConSol CM-Groovy-API komplizierte Workflows zu erstellen.

Bitte wenden Sie sich an den CM-Vertrieb, wenn Sie an dem von ConSol angebotenen Groovy-Training **Groovy in a Nutshell** interessiert sind.

Im vorliegenden Handbuch verwenden wir Java-Stil und Groovy-Stil. Sie können entscheiden, welchem Stil Sie folgen möchten.

D.1.1 Einige kurze Beispiele für die Programmierung im Java- vs. Groovy-Stil

Wie oben erwähnt, müssen Sie für ConSol CM-Skripte Groovy verwenden. Der gleiche Inhalt kann möglicherweise auf unterschiedliche Art ausgedrückt oder programmiert werden. Die folgenden Abschnitte enthalten einige Hinweise und Beispiele für die Arbeit mit der Groovy-API.

D.1.1.1 Getter-Methoden können häufig ausgelassen werden

Die meisten Groovy-Objekte haben mehrere *Getter-Methoden*, mit denen Werte aus Objektattributen abgerufen werden. Sie können entweder die vollständigen *Getter-Methoden* oder die Kurzform verwenden. Siehe auch folgende Beispiele für Workflow-Skripte.

Anwendungsfall	Java-gemäße Syntax (Langform)	Groovy-Syntax (Kurzform)
Ticketthema abrufen	`String mysubject = ticket.getSubject()`	`def mysubject = ticket.subject`
Bearbeiter des Tickets abrufen	`Engineer myeng = ticket.getEngineer()`	`def myeng = ticket.engineer`
Hauptkontakt des Tickets abrufen	`Unit mymaincontact = ticket.getMainContact()`	`def mymaincontact = ticket.mainContact`
Wert eines bestimmten Benutzerdefinierten Feldes aus dem Ticket abrufen	`String myprio = ticket.get ("helpdesk_fields", "prio")`	`def myprio = ticket.get ("helpdesk_fields.prio")`

Anwendungsfall	Java-gemäße Syntax (Langform)	Groovy-Syntax (Kurzform)

Unit-Typ des Haupt-kontakts abrufen	`Unit mycustomer = workflowApi.getPrimaryContact()`
	`UnitDefinition myunitdef = mycustomer.getDefinition()`
	`UnitDefinitionType mydeftype = myunitdef.getType()`
		`def mycustomer = workflowApi.primaryContact`
		`def myunitdef = mycustomer.definition`
		`def mydeftype = mycustomer.definition.type`

Der Zugriff auf Benutzerdefinierte Felder kann nicht abgekürzt werden, da es für diese Felder keine getter-Methoden gibt. Details über die Arbeit mit Daten aus Benutzerdefinierten Feldern finden Sie im Abschnitt Arbeiten mit Datenfeldern.

D.1.1.2 Setter-Methoden können häufig ausgelassen werden

Die meisten Groovy-Objekte haben mehrere *Setter-Methoden*, mit denen Werte für Objektattribute gesetzt werden. Sie können entweder die vollständigen *Setter-Methoden* oder die Kurzform verwenden. Siehe auch folgendes Beispiel für Workflow-Skripte.

Anwendungsfall	Java-gemäße Syntax (Langform)	Groovy-Syntax (Kurzform)
Ticketthema setzen	`ticket.setSubject("asd")`	`ticket.subject = "asd"`

D.2 CM API-Dokumentation

Für die ConSol CM-API steht eine Groovy API Doc zur Verfügung. Fragen Sie Ihren ConSol CM-Consultant oder -Vertriebsmitarbeiter nach der entsprechenden .jar-Datei.

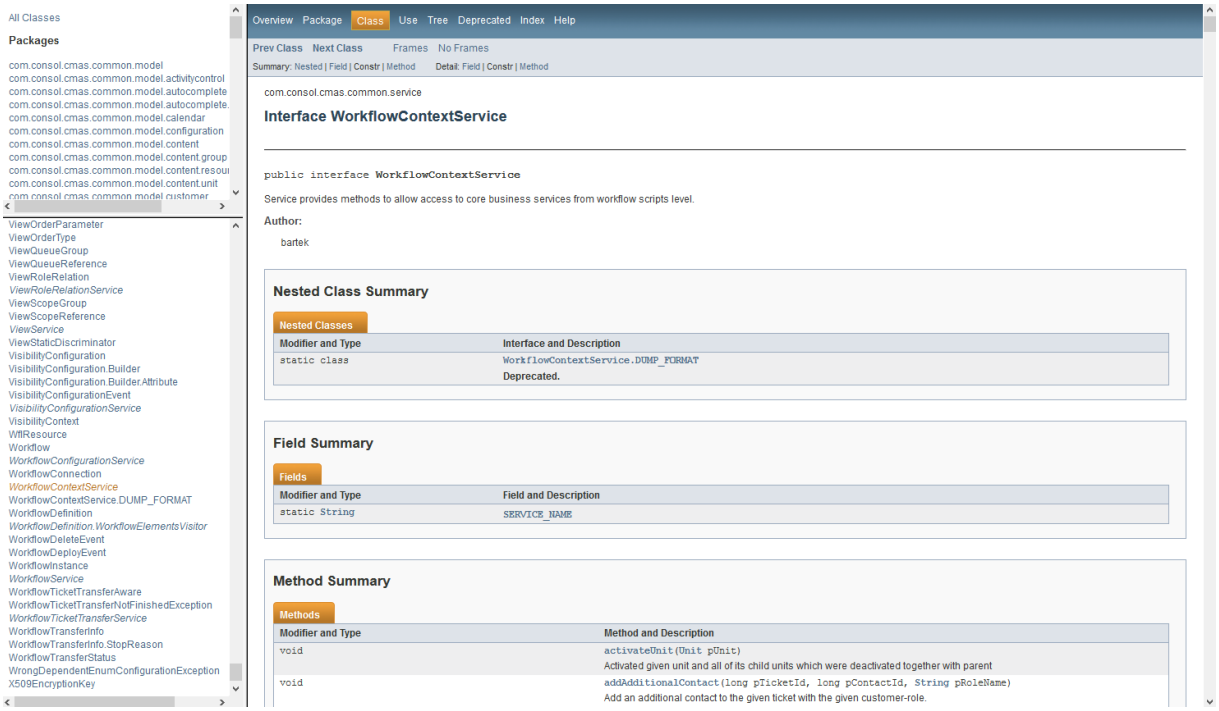


Abbildung 113: *ConSol CM Groovy API Doc*

D.3 CM-Skripttypen in Workflows

In ConSol CM-Workflows werden Skripte in folgenden Kontexten verwendet:

- Als Aktivitätsskript für eine Aktivität.
- Als Bedingungsskript für eine Aktivität, das *true* oder *false* zurückgeben muss.
- Als Skript für einen Entscheidungsknoten, das *true* oder *false* zurückgeben muss.
- Als Skript für einen Event-Trigger, das ausgeführt werden muss, bevor der Trigger feuert.
- Als Skript für einen Zeit-Trigger,
    - das ausgeführt wird, wenn der Zeit-Trigger initialisiert wird, d. h. wenn das Ticket in den Bereich wandert, an dem der Zeit-Trigger hängt.
    - das ausgeführt wird, wenn der Zeit-Trigger feuert, d. h. wenn die definierte Zeit abgelaufen ist.
- Als Skript für Endknoten.
- Als Skript für Einsprung- und Aussprungknoten.
- Als Bedingungsskripte für ACFs, die *true* oder *false* zurückgeben müssen.
- Als Initialisierungsskripte für ACFs.

Eine Erklärung über das Einfügen dieser Skripte finden Sie in den entsprechenden Abschnitten dieses Handbuchs.

D.4 Zusammenspiel von Skripten

Für jedes Workflow-Skript können Sie auswählen, ob der Code direkt im Workflow ausgeführt werden soll, oder ob das Skript im Admin Tool im Abschnitt *Skripte* gespeichert werden soll und aus dem Workflow-Skript aufgerufen werden soll. Siehe Abschnitt Speichern einiger Workflow-Skripte im Admin Tool.

D.5 Skripte in ConSol CM im Allgemeinen

Denken Sie daran, dass die Konfiguration und Programmierung mit dem Process Designer nur die "Hälfte der Intelligenz" Ihres ConSol CM-Systems darstellt. Viele Konfigurationen und Skripte werden im Admin Tool verwaltet. Informationen zu den Skripten finden Sie im Abschnitt über die Admin-Tool-Skripte des *ConSol CM Administratorhandbuchs*.

D.6 Prozesslogik

In diesem Kapitel werden folgende Themen behandelt:

D.6.1 Einleitung

Wenn Sie Workflows erstellen und ändern, ist es wichtig, die Grundprinzipien der Workflow-Engine zu kennen, die das Verhalten von Tickets während des Prozesses bestimmen. Deshalb erhalten Sie hier eine kurze Übersicht über die Grundregeln der Ticketverarbeitung in ConSol CM.

D.6.2 Aktivitäten

Grundregeln:

- Beim Durchlaufen eines Workflows wartet das Ticket immer **hinter** der letzte Aktivität und **nicht** vor der nächsten!
- Danach schaut es, welche Aktivität als nächstes ausgeführt/durchlaufen werden kann.
- Wenn die nächste mögliche Aktivität eine manuelle Aktivität ist, bleibt das Ticket an der Position nach der vorherigen Aktivität (Nummer **(1)** und **(2)** in der folgenden Abbildung).
- Wenn die nächste mögliche Aktivität eine automatische Aktivität ist, wird die Aktivität ausgeführt, d. h. das Ticket durchläuft diese Aktivität (Nummer **(3)** in der folgenden Abbildung).
- Eine Aktivität kann **eine oder mehrere manuelle** Folgeaktivitäten haben **oder** eine Aktivität kann (nur) **eine automatische** Aktivität als Folgeaktivität haben.
- Wenn Sie einen Workflow speichern, führt der Process Designer automatisch eine Konsistenzprüfung durch. Wenn Inkonsistenzen festgestellt werden (z. B. zwei automatische Aktivitäten, die auf dieselbe Vorgängeraktivität folgen), wird eine Fehlermeldung angezeigt und der Workflow kann nicht gespeichert werden.

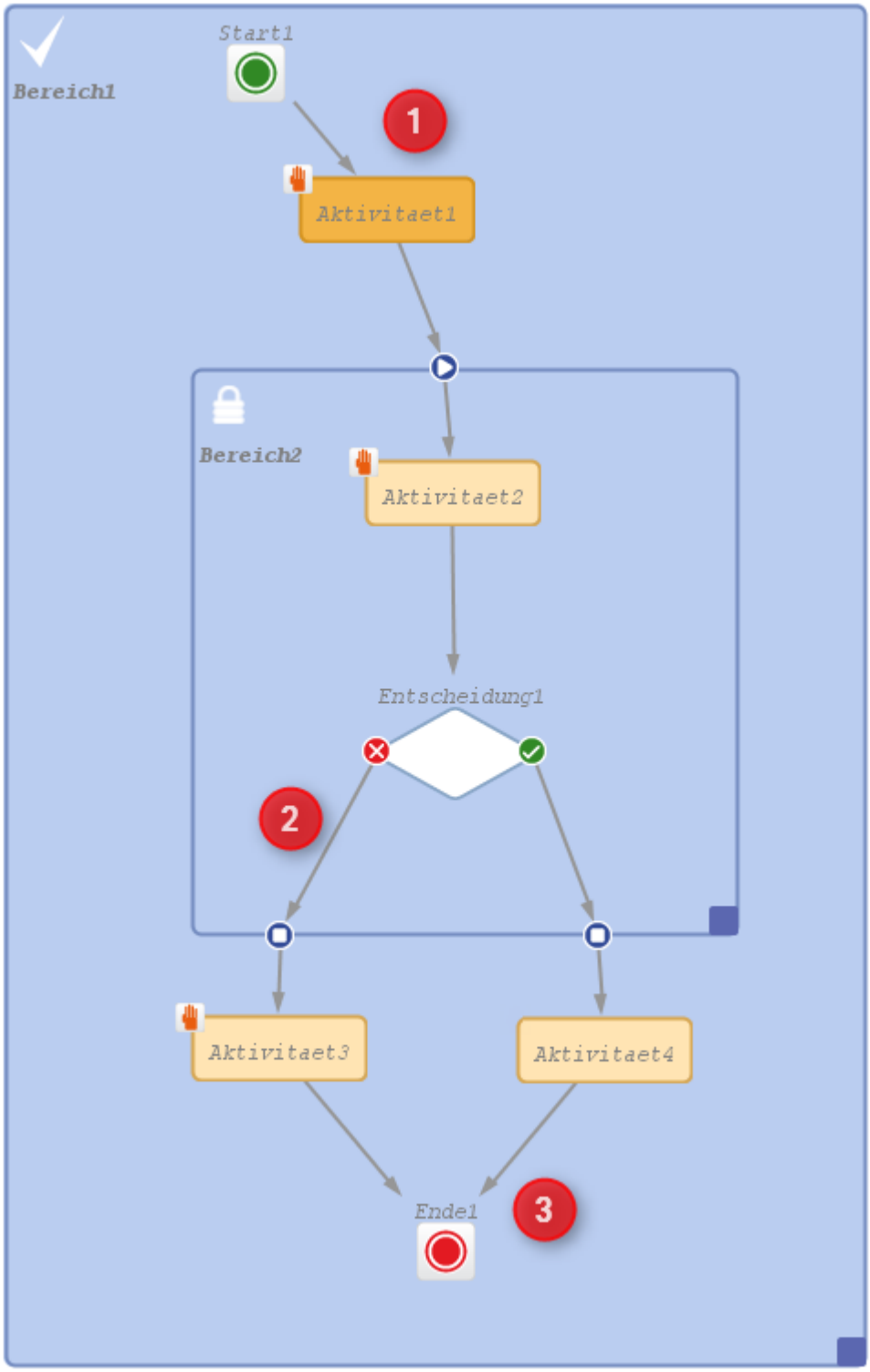


Abbildung 114: *ConSol CM Process Designer - Prozesslogik 1*

D.6.3 Interrupts und Exceptions

Im Prozessverlauf, d. h. während das Ticket offen ist und die Bearbeiter daran arbeiten, können Ereignisse eintreten, um die sich ein Mitarbeiter kümmern muss. Wenn das Ticket zum Beispiel eine E-Mail erhält oder wenn eine Zeitspanne für eine SLA abgelaufen ist, ist es wichtig, dieses Ereignis zu erfassen und entsprechend zu reagieren.

Die Reaktion und das Verhalten der Tickets können auf zwei Wegen definiert werden. Sie können Folgendes implementieren ...

- **Interrupt**
  Dies ist eine Workflow-Architektur, bei der das Ereignis erfasst wird, eine oder mehrere Aktivitäten ausgeführt werden, und das Ticket an seine vorherige Position im Workflow zurückkehrt.
- **Exception**
  Dies ist eine Workflow-Architektur, bei der das Ereignis erfasst wird und aufgrund der folgenden manuellen oder automatischen Aktivitäten seine vorherige Position verlässt und eine neue Position innerhalb des Workflows oder in einem anderen Workflow einnimmt.

D.6.3.1 Interrupts

Interrupts ...

- werden durch Trigger aktiviert.
- führen zu einer kurzen Unterbrechung des Prozesses, um auf das Ereignis des Triggers zu reagieren.
- verwenden automatische Aktivitäten (eine oder mehrere aufeinanderfolgende automatische Aktionen).
- bringen das Ticket an seine vorherige Position im Workflow zurück, d. h. an die Position, an der sich das Ticket befand, als das Interrupt-Ereignis gefeuert hat.
- werden häufig verwendet, um das Ticket-Icon mit einem Overlay zu markieren, z. B. wenn eine E-Mail eingegangen ist (siehe Abbildung unten) oder wenn eine Eskalationszeit erreicht wurde.

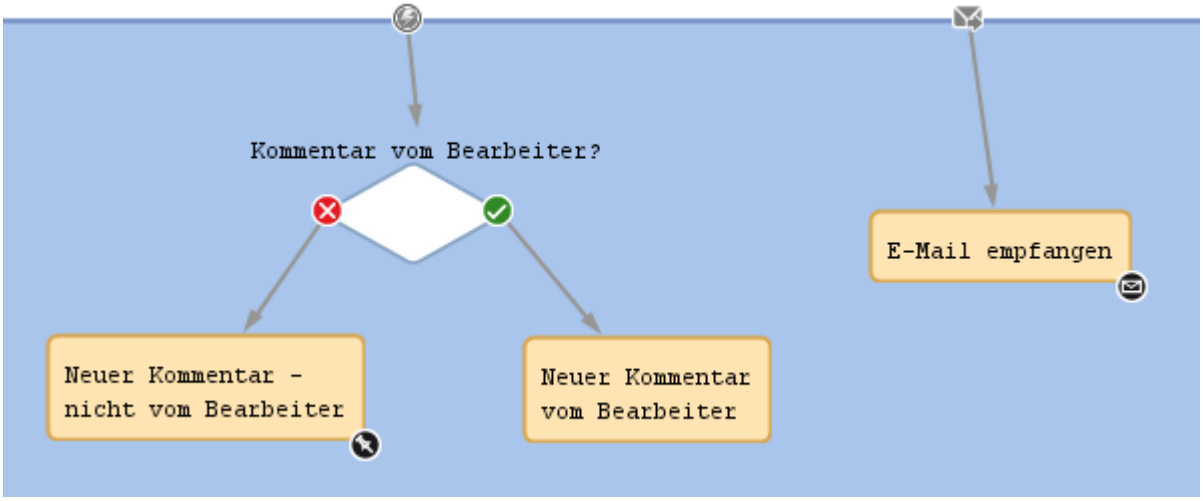


Abbildung 115: *ConSol CM Process Designer - Zwei Interrupts*

D.6.3.2 Exceptions

Exceptions ...

- werden durch Trigger aktiviert.
- verschieben das Ticket von seiner alten Position im Workflow an eine neue Position. Letztere kann sich im gleichen oder in einem anderen Workflow befinden.
- führen dazu, dass der Prozess an der neuen Position fortgesetzt wird.

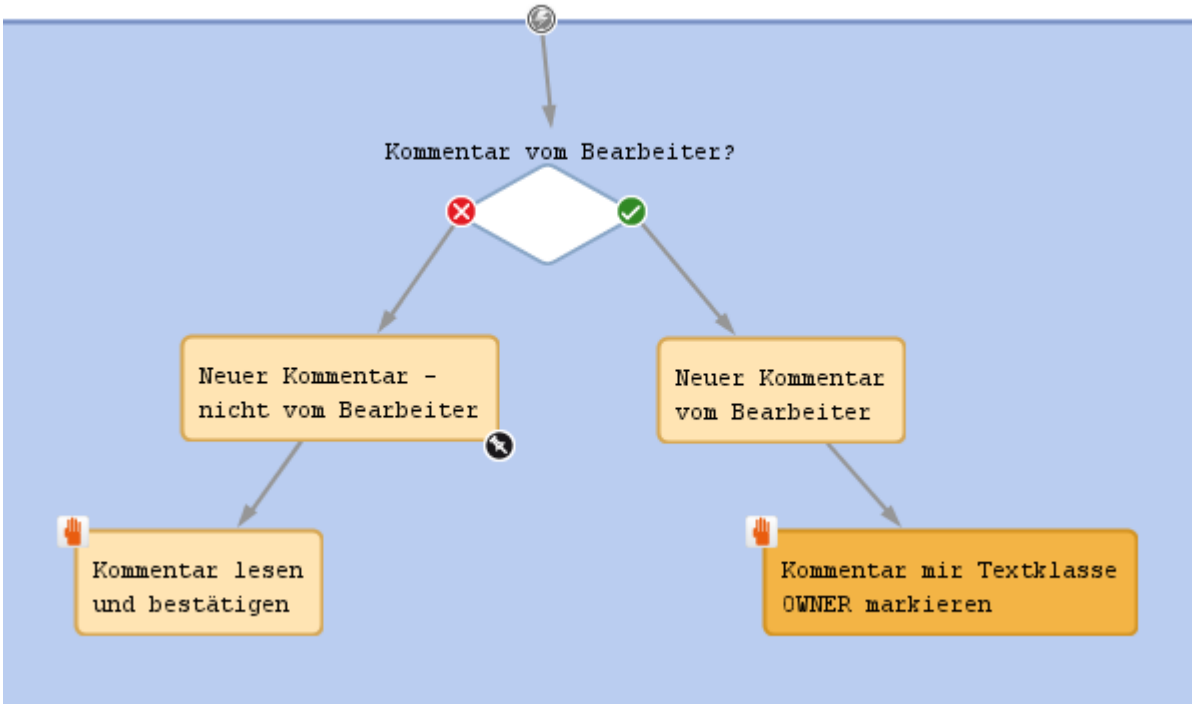


Abbildung 116: *ConSol CM Process Designer - Exception*

D.6.4 Schleifen (Fehler in Workflows)

(Endlos-)Schleifen führen zu Fehlern im Prozess. Sie werden vom Process Designer nicht erkannt. Es ist also möglich, einen Workflow zu installieren, der eine Schleife enthält, wie in der folgenden Abbildung gezeigt.

Die Prozess-Engine erkennt solche Schleifen aber zur Laufzeit und wirft eine *InfiniteWorkflowLoopException*, um einen vollständigen Systemausfall zu verhindern. Sie können diese Exception in der Datei *server.log* sehen. Im Web Client wird eine Fehlermeldung angezeigt.

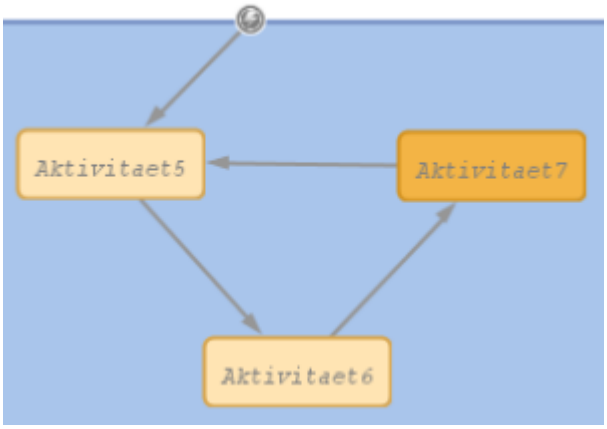


Abbildung 117: *ConSol CM Process Designer - Schleife im Workflow*

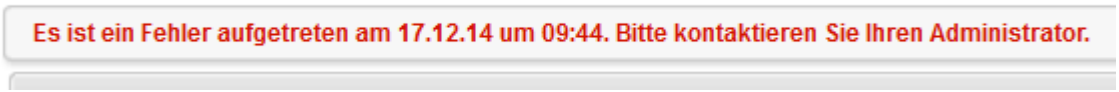


Abbildung 118: *ConSol CM Web Client - Fehlermeldung beim Erkennen der Schleife*

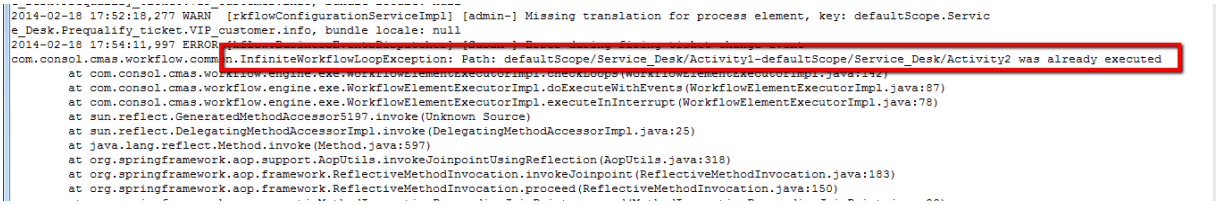


Abbildung 119: *Konsole - Datei server.log: Von Workflow-Schleife hervorgerufene Fehlermeldung*

Event-Trigger können ebenfalls Schleifen hervorrufen, wenn die automatische Aktivität nach dem Trigger den Parameter ändert, auf den der Trigger reagiert. Siehe Abschnitt Vermeiden von sich selbst auslösenden Event-Triggern.

D.6.5 Prozesslogik von Zeit-Triggern

Siehe Abschnitt Geschäftslogik und Initialisierung eines Zeit-Triggers.

D.6.6 Prozesslogik von Event-Triggern

Siehe Abschnitt Geschäftslogik von Event-Triggern .

D.7 Wichtige Klassen und Objekte

In diesem Kapitel werden folgende Themen behandelt:

D.7.1 Einleitung

Zur Erleichterung der Programmierung von ConSol CM-Skripten bietet die CM-Workflow-API einfachen Zugriff auf die häufig verwendeten Objekte. Zudem können verschiedene Objekte und Methoden über Convenience-Klassen und -Methoden auf kurzem Weg aufgerufen werden.

D.7.2 Wichtige Objekte

Einige Objekte sind in Workflow-Skripten implizit verfügbar.

Die gleichen Objekte sind in Admin-Tool-Skripten nicht implizit verfügbar, d. h. in Admin-Tool-Skripten müssen Sie für die entsprechenden Klassen oder Pakete *Import-Anweisungen* verwenden!

D.7.2.1 Ticket

In jedem Workflow-Skript kann man über das Objekt *ticket* einfach auf das aktuelle Ticket zugreifen. Es leitet sich aus der Klasse *Ticket* ab und ist implizit verfügbar. Es sind kein Import und keine Instanziierung erforderlich.

Beispiel:

```
def myId = ticket.getId()
//or shorter, Groovy-like:
def myId = ticket.id
```

Code-Beispiel 13: *Verwenden des Ticket-Objekts*

D.7.2.2 workflowAPI

Das Objekt *workflowApi* ist ebenfalls implizit vorhanden. Es bietet einfachen Zugriff auf das Interface *WorkflowContextService*, das für viele Operationen verwendet wird.

Beispiele:

```
workflowApi.sendEmail(contact_e,subj,text,replyto,null)
```

Code-Beispiel 14: *Verwenden von workflowApi zum Senden einer E-Mail (Ältere Variante. In der aktuellen Variante wird ein Objekt der Klasse Mail verwendet.)*

```
def curr_eng = workflowApi.getCurrentEngineer()
ticket.setEngineer(curr_eng)
```

Code-Beispiel 15: *Verwenden von workflowApi zum Zuweisen eines Tickets an den aktuellen Bearbeiter*

```
workflowApi.deactivateTimer("defaultScope/Service_Desk/TimeTrigger1")
```

Code-Beispiel 16: *Verwenden von workflowApi zum Deaktivieren eines Triggers*

```
workflowApi.addValidationError("1", "The ticket cannot be closed before a solution
 is provided. Please fill-in solution and mark it with text class SOLUTION first.")
```

Code-Beispiel 17: *Verwenden von workflowAppi zum Anzeigen einer GUI-Meldung für den Bearbeiter*

D.7.2.3 trigger

Dieses Objekt ist implizit in Skripten von Zeit-Triggern (*Skript zu Beginn*, *Skript nach Ablauf*) verfügbar.

```
def addedEscalMillis = 0
switch (ticket.queue.name) {
   case "HelpDesk_1st_Level":
      addedEscalMillis = 12*60*60*1000L;
      break;
   case "HelpDesk_2nd_Level":
      addedEscalMillis = 24*60*60*1000L;
      break;
   case "ServiceDesk":
      addedEscalMillis = 4*60*60*1000L;
   }
trigger.setDueTime(addedEscalMillis)
```

Code-Beispiel 18: *Beispiel für ein Skript zu Beginn*

D.7.3 Convenience-Klassen und Methoden

Die ConSol CM-API bietet verschiedene Convenience-Interfaces und -Methoden, die den Zugriff auf die meisten Objekte der alltäglichen CM-Programmierung deutlich vereinfachen. Die meisten dieser Convenience-Interfaces gehören zum Paket *com.consol.cmas.common.service* und seinen Unterpaketen. Details dazu finden Sie in der *ConSol CM-Java-API-Dokumentation*. An dieser Stelle zeigen wir Ihnen einige Beispiele, die für die meisten CM-Programmierer nützlich sein können.

Die implementierende Instanz des Interface ist immer verfügbar, indem Sie den ersten Buchstaben (ein Großbuchstabe) im Klassennamen durch einen Kleinbuchstaben ersetzen, z. B. das Objekt (Singleton) mit dem Interface *EngineerService* ist mit dem Objekt *engineerService* verfügbar, siehe *Beispiel 2*.

D.7.3.1 Beispiel 1: Verwenden von ConfigurationService zum Abrufen von System-Properties

```
def tic_nr = configurationService.getValue("custom-mycompany-properties","engineer_
 management.ticket.nr")

// then: ... do something with the engineer management ticket,
// e.g. find out the name of the next engineer a service ticket
//should be assigned
```

Code-Beispiel 19: *Verwenden von ConfigurationService zum Abrufen der Nummer des Bearbeiterverwaltungs-Tickets*

```
def baseUrl = configurationService.getValue("custom-mycompany-
 properties","base.url.mycompany")
def url = baseUrl + "/cm-client/ticket/ticket_name/" + ticket.getName()
def itComplete = url + " " + ticket.getName()
//alternative: def itComplete = "${url} ${ticket.name}"

// ... do something with the ticket url, e.g. place a link to a child ticket in a
 table of the parent ticket
```

Code-Beispiel 20: *Verwenden von ConfigurationService zum Abrufen der Basis-URL des Systems*

D.7.3.2 Beispiel 2: Verwenden von EngineerService zum Zuweisen des Tickets an einen Genehmiger

```
// Script does the following:
// Hand-over ticket to approver only when approver has been set in ticket as
 additional engineer
// Import package, because classes are not available in workflow otherwise:
import com.consol.cmas.common.model.ticket.user.function.*

// Get the name of the approver which has been written/stored in a Custom Field, //
 namely the field with the name
// CF_ApproverName in the Custom Field Group CF_GroupApproverData. The value could
 be for example Mr. Miller:
def gen = ticket.get("CF_GroupApproverData.CF_ApproverName").getName()

// Get the engineer object where the name Mr. Miller is set, i.e.
// the engineer object of the desired approver:
def gen_eng = engineerService.getByName(gen)

// Get the ticketFunction object which represents the ticketFunction (engineer
 role) Approver:
TicketFunction tf = ticketFunctionService.getByName("Approver")

// Add the engineer object of Mr. Miller as Approver. i.e.
// in the ticketFunction (engineer role) Approver to the ticket.
// One of the paramaters is ticket. This does not have to be instantiated,
// because it is implicitly present in workflow
// scripts:
def tu = ticketUserService.addTicketUser(ticket, gen_eng, tf, "Approver")

// Assign the ticket to the engineer, i.e. set the engineer Mr. Miller also as
 ticket owner.
def tic2 = workflowApi.assignEngineer(ticket, gen_eng)
```

Code-Beispiel 21: *Verwendung von EngineerService*

Es gibt zwei Zuweisungen:

1. Hr. Miller wird als zusätzlicher Bearbeiter in der Bearbeiterfunktion *Genehmiger* gesetzt.
2. Hr. Miller wird als zugewiesener Bearbeiter des Tickets gesetzt.

D.7.3.3 Beispiel 3: Verwenden von EnumService zum Abrufen eines Enum-Werts über den Namen

```
def enumValueMLA = enumService.getValueByName( "priority", "REGULAR" )
ticket.set( "helpdesk_fields.prio", enumValueMLA )
```

Code-Beispiel 22: *Verwenden von EnumService zum Abrufen eines Enum-Werts über den Namen*

D.7.3.4 Beispiel 4: Verwenden von TicketService zum Abrufen aller Tickets einer bestimmten Sicht

```
List<Ticket> mylist = ticketService.getByView(new ViewCriteria(
   viewService.getByName("helpdesk_active_tickets"),
   ViewAssignmentParameter.allAssignedTickets(),
   ViewGroupParameter.allTickets(),
   viewOrderParameter.addByName(true)))
```

Code-Beispiel 23: *Verwenden von TicketService zum Finden der Tickets einer Sicht*

D.7.3.5 Beispiel 5: Verwenden von EngineerRoleRelationService zum Senden einer E-Mail an alle Bearbeiter mit einer Rolle

```
// Send e-mail to all engineers of a regular role

def mail = new Mail()
mail.setTo(engineerRoleRelationService.getEngineersWithRoles(roleService.getByName
  ("Supervisor"))*.email.join(","))
mail.setSubject("Ticket (${ticket.name}) -- Escalation!")
mail.setText(workflowApi.renderTemplate("Ticket escalation note to supervisor"))
mail.send()
```

Code-Beispiel 24: *Verwenden von EngineerRoleRelationService zum Senden einer E-Mail an alle Bearbeiter einer Rolle*

D.8 Arbeiten mit Datenfeldern

In diesem Kapitel werden folgende Themen behandelt:

D.8.1 Einführung in Datenfelder

Der Zugriff auf die Datenfelder ist ein wesentlicher Bestandteil der ConSol CM-Programmierung. Er kann in allen Skripten des Systems notwendig sein, sowohl in Workflow-Skripten als auch in Admin-Tool-Skripten und zwar unabhängig vom Skripttyp. An dieser Stelle konzentrieren wir uns auf die Workflow-Programmierung, der Zugriff auf die Datenfelder funktioniert aber in allen Skripten grundsätzlich gleich.

D.8.1.1 ConSol CM-Version 6.9 und höher

Ab ConSol CM-Version 6.9.0 gibt es zwei Arten von Datenfeldern:

- **Benutzerdefinierte Felder**
  Werden zur Definition von Ticketdaten verwendet und in Benutzerdefinierten Feldgruppen verwaltet. Aus früheren CM-Versionen bekannt.
- **Datenobjektgruppenfelder**
  Werden zur Definition von Kundendaten in *FlexCDM*, dem neuen Kundendatenmodell, verwendet und in Datenobjektgruppen verwaltet.

Die Arbeit mit Datenfeldern des neuen Kundendatenmodells (FlexCDM) in Version 6.9 und höher ist im *ConSol CM Administratorhandbuch - Kundendatenmodell 6.9: FlexCDM* und *ConSol CM Administratorhandbuch (Version 6.9)*, Abschnitt *Das CM-Kundendatenmodell: FlexCDM* detailliert beschrieben.

Regeln für die Arbeit mit Datenfeldern in CM 6.9 und höher:

Wenn Sie mit Benutzerdefinierten Feldern und Datenobjektgruppenfeldern arbeiten, sollten Sie drei Grundregeln berücksichtigen:

1. Benutzerdefinierte Felder werden immer in Benutzerdefinierten Feldgruppen verwaltet und referenziert, d. h. wenn Sie den Wert eines Benutzerdefinierten Feldes abrufen wollen, müssen Sie *<Name Benutzerdefinierte Feldgruppe>.<Name Benutzerdefiniertes Feld>* verwenden.
2. Datenobjektgruppenfelder werden immer in Datenobjektgruppen verwaltet und referenziert, d. h. wenn Sie den Wert eines Datenobjektgruppenfeldes abrufen wollen, müssen Sie *<Name Datenobjektgruppe>.<Name Datenobjektgruppenfeld>* verwenden.
3. Sie müssen immer den eindeutigen technischen Namen verwenden, um ein Datenobjektgruppenfeld oder eine Datenobjektgruppe zu referenzieren, nie den lokalisierten Wert.

D.8.1.2 ConSol CM-Version 6.10 und höher

In ConSol CM-Version 6.10 wurde das Modul CM.Resource Pool eingeführt. Eine detaillierte Beschreibung aller Objekte im neuen Modul finden Sie im *ConSol CM Administratorhandbuch*, Abschnitt *CM.Resource Pool*. Zu den bereits aus Version 6.9 bekannten Datenfeldern kommt eine neue Art von Datenfeldern: die Ressourcenfelder. In CM-Versionen 6.10 und höher arbeiten wir mit folgenden Datenfeldern:

- **Benutzerdefinierte Felder**
  Werden zur Definition von Ticketdaten verwendet und in Benutzerdefinierten Feldgruppen verwaltet. Aus früheren CM-Versionen bekannt.
- **Datenobjektgruppenfelder**
  Werden zur Definition von Kundendaten in *FlexCDM*, dem Kundendatenmodell, verwendet und in Datenobjektgruppen verwaltet.
- **Ressourcenfelder**
  Werden zur Definition von Ressourcendaten im Ressourcendatenmodell verwendet und in Ressourcenfeldgruppen verwaltet.

D.8.2 Datentypen für Datenfelder

Ein Datenfeld hat immer einen bestimmten Datentyp. Bei Variablen für die Programmierung hängt es vom Datentyp ab, wie Sie den Wert des Feldes verarbeiten müssen, ein Feld des Datentyps *string* kann z. B. nicht für die Berechnung von Zahlen verwendet werden und Felder des Datentyps *enum* benötigen eine spezielle Zugriffsmethode.

Die folgenden Datentypen sind für Benutzerdefinierte Felder, Datenobjektgruppenfelder und Ressourcenfelder verfügbar.

- **boolean (Ja/Nein)**
  Werte: true/false. Je nach Einstellung in der Annotation *boolean-type* wird der Wert als Checkbox, Radio-Button oder Drop-down-Liste angezeigt.

  Wenn Sie in CM-Workflows oder im Admin Tool mit Skripten arbeiten, sollten Sie wissen, dass das Verhalten von Boolean-Feldern, die als Checkboxen dargestellt werden, d. h. mit der Annotation *boolean-type = checkbox* (Standardwert) versehen sind, anders ist als in früheren CM-Versionen!

  - **In CM-Versionen vor 6.9.4.0:**
    Wenn ein Boolean-Feld noch nicht angefasst wurde, ist der Wert *false*.
    Wenn es aktiviert wird, ist der Wert *true*, und wenn es danach wieder deaktiviert wird, ist der Wert wieder *false*.
  - **In Version 6.9.4.0 und höher:**
    Wenn ein Boolean-Feld noch nicht angefasst wurde, ist der Wert *NULL*.
    Wenn es aktiviert wird, ist der Wert *true*, und wenn es danach wieder deaktiviert wird, ist der Wert *false*.

  Felder, die in der Datenbank bereits mit Werten gefüllt sind, werden während des Updates von einer Version vor 6.9.4.0 auf Version 6.9.4.0 und höher nicht verändert.

  Boolean-Felder, die als Radio-Button (Annotation *boolean-type = radio*) oder Drop-down-Liste (Annotation *boolean-type = select*) dargestellt sind, hatten schon immer das Verhalten, das für Version 6.9.4.0 oder höher beschrieben ist, d. h. sie haben den Wert *NULL*, wenn sie noch nicht angefasst wurden.

- **date (Datum)**
  Format und Genauigkeit können über Annotationen festgelegt werden.

- **enum (Sortierte Liste)**
  Für Sortierte Listen. Die Bearbeiter können im Web Client zwischen einem der Listenwerte wählen. Die sortierten Listen und ihre Werte müssen zuvor in der Enum-Verwaltung im Admin Tool erstellt worden sein. Wählen Sie den gewünschten *Listentyp* und die *Listengruppe* in den Feldern darunter.
- **list (Liste)**
  Ein Feld dieses Datentyps ist der erste Schritt zur Erstellung einer Liste (eine Spalte) oder einer Tabelle (mehrere Spalten) von Eingabefeldern im Web Client.
  - Für eine **Tabelle** wird im nächsten Schritt ein anderes Feld des Typs *struct* erstellt (siehe unten), das später die Eingaben der einzelnen Listenfelder enthält (die die Spalten der Tabelle sind). Wenn Sie also eine Tabelle erstellen möchten, müssen Sie zuerst ein Feld des Typs *struct* erzeugen (siehe unten), bevor Sie die Felder für die Tabellenspalten hinzufügen können.
  - Für eine **einfache Liste** ist der nächste Schritt die Erstellung der Felder, die zu der Liste gehören. In diesem Fall ist kein *struct* erforderlich.

  Für alle Felder, die zu einer Liste oder Tabelle gehören, müssen Sie die Abhängigkeiten im Feld *Gehört zu* angeben (siehe unten). Ein Tabellenfeld (ein normales Datenfeld), gehört zum Beispiel immer zu einem *struct* und ein *struct* gehört immer zu einer *list*.
- **struct (Struktur)**
  Mit einem Datenfeld dieses Typs wird eine Datenstruktur (Zeile einer Tabelle) definiert, die ein oder mehrere Felder umfasst. Das ist der zweite Schritt zur Erstellung einer Tabelle, nachdem Sie zuerst ein Feld des Typs *list* erstellt haben. Im nächsten Schritt fügen Sie die Felder für die Spalten der Tabelle hinzu. Die Abhängigkeiten müssen für jedes Datenfeld im Feld *Gehört zu* (siehe unten) gesetzt werden, d. h. ein *struct* gehört immer zu einer *list*.

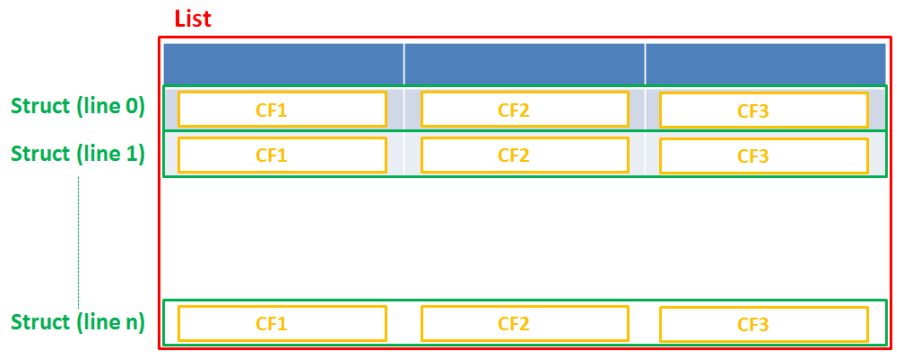


Abbildung 120: *Schema: List of Structs*

Technisch gesehen ist eine Liste ein Array, der in jedem Feld eine Map (= Paare von Schlüssel:Wert) enthält.

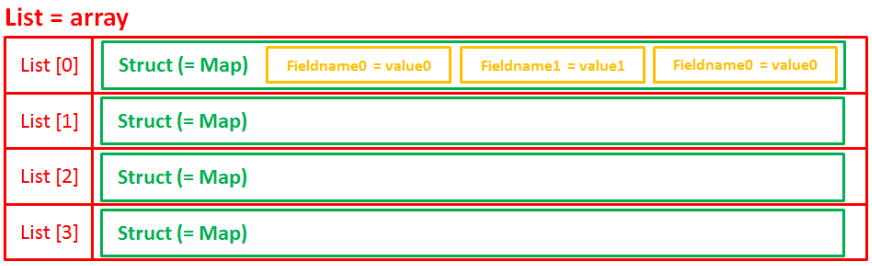


Abbildung 121: *List of structs, technisches Prinzip*

- **number (Zahl)**
  Für Ganzzahlen (Integer).
- **fixed point number (Festkommazahl)**
  Für Festkommazahlen, z. B. Währungen. Sie müssen die Gesamtanzahl der Ziffern (*Genauigkeit*) und die Anzahl der Ziffern nach dem Komma (*Skalierung*) in den entsprechenden Feldern eingeben.
- **string (Text)**
  Für bis zu 4000 alphanumerische Zeichen.

  ⚠ Beschränkung bei der Verwendung einer Oracle-Datenbank: Es können höchstens 4000 Bytes in UTF-Codierung gespeichert werden. Ab Oracle12c.

- **long string (Text)**
  Für große Objekte

  ⚠ Beim Datentyp *long string (Text)* hängt die Grenze von dem für ConSol CM verwendeten Datenbanksystem ab: MS SQL Server: 2 GByte; MySQL: 4 GByte; Oracle: 16 - 64 GByte (je nach Page-Größe des Tablespace).

- **short string (Text)**
  Für bis zu 255 alphanumerische Zeichen.

  ⓘ Bei Feldern des Datentyps *string (Text)* können Sie die Felddefinition mithilfe von Annotationen präzisieren. Ein Feld des Datentyps *string* kann zum Beispiel so definiert werden, dass es eine URL enthält und automatisch als Hyperlink angezeigt wird. Lesen Sie dazu den folgenden Abschnitt.

- **contact data reference (Referenz auf ein Kontaktdatenfeld)**
  Besonderer Datentyp, der intern als Referenz auf die mit dem Ticket verknüpften Kontakte verwendet wird. In FlexCDM (d. h. ab CM-Version 6.9.0), wird dieser Datentyp nicht mehr angezeigt, sondern nur noch intern im CM-System verwendet.

- **MLA field (Baum sortierter Listen)**
  Dieser Datentyp wird für Felder verwendet, die hierarchische Listen mit einer Baumstruktur enthalten, das sogenannte *MLA* (Multi Level Attributes). Der Name dieses Feldes ist der Name des neuen MLA, das im Admin Tool in der MLA-Verwaltung angelegt werden muss. Die Gruppe des Feldes muss angegeben werden, nachdem das MLA erstellt wurde.

⚠ Der Datentyp, den Sie beim Erstellen eines Datenfeldes verwenden, kann danach nicht mehr verändert werden!

D.8.3 Details über String-Felder: Verwenden von Annotationen zum Anpassen von Strings

String-Felder werden häufig für Kunden-, Ticket- und Ressourcendaten verwendet und in Strings kann unterschiedlicher Inhalt gespeichert werden, zum Beispiel ein Textfeld mit einem Kommentar, ein einfaches Eingabefeld mit nur 20 Zeichen, eine URL oder ein Passwort. Das Anpassen von String-Feldern erfolgt mithilfe von speziellen Annotationen, die auf der Seite Annotationen des Anhangs aufgeführt sind. Da die Arbeit mit diesen Annotationen zu den alltäglichen Aufgaben eines CM-Administrators gehört, sind die wichtigsten und am häufigsten verwendeten Annotationen zusätzlich an dieser Stelle erklärt.

Wie kann ich ...

... statt einer einzelnen Zeile ein **Textfeld** einfügen?

Wert für Annotation *text-type*: **textarea**

Die Größe des Textfeldes kann angepasst werden. Es wird je nach Webbrowser als Standardtextfeld angezeigt. Verwenden Sie die Annotation *field-size*, wenn das Textfeld eine bestimmte Größe haben soll.

... die Eingabe des Feldes für **Passwörter** ausblenden?

Wert für Annotation *text-type*: **password**

Es werden nur Punkte angezeigt. Diese Annotation legt **nicht** fest, dass das Feld ein Passwort enthält! Es definiert nur den Anzeigemodus! Verwenden Sie die Annotation *password*, um ein String-Feld zu definieren, das das CM.Track-Passwort enthält.

... einen **Hyperlink** anzeigen, den Namen statt dem Link anzeigen?

Wert für Annotation *text-type*: **url**

Die Eingabe wird im *Ansichtsmodus* als Hyperlink angezeigt. Der String muss einem bestimmten URL-Muster entsprechen:

- "^((?:mailto\:|(?:(?:ht|f)tps?)\://)1\S+)(?: (?:\|)?(.*))?$"

Der erste Teil des Strings ist der Link (URL) und der zweite Teil ist der Name, der angezeigt werden soll.

Beispiel: "http://consol.de ConSol"

... einen **Dateilink** anzeigen?

Wert für Annotation *text-type*: **file-url**

Die Eingabe wird als Link auf eine Datei im Dateisystem angezeigt. Der Webbrowser muss solche Links zulassen/unterstützen!

Beispiel: Aktivieren von file://-URLs in einem Firefox-Browser

Fügen Sie die folgenden Zeilen entweder zur Konfigurationsdatei *prefs.js* oder zu *user.js* im Benutzerprofil hinzu. Auf einem Windows-System befindet sich diese normalerweise in einem Ordner wie `C:\Benutzer\<BENUTZERNAME>\AppData\Roaming\Mozilla\Firefox\Profiles\uvubg4fj.default`

- user_pref("capability.policy.localfilelinks.checkloaduri.enabled", "allAccess");
- user_pref("capability.policy.localfilelinks.sites", "http://cm-server.domain.com:8080");
- user_pref("capability.policy.policynames", "localfilelinks");

Alternativ kann ein Add-on des Firefox-Browsers wie *Local Filesystem Links* installiert werden, um einfacher auf die referenzierten Dateien und Ordner zugreifen zu können.

Dieser Link wird auch als Tooltip angezeigt.

Die URL ist korrekt formatiert, wenn folgende Bedingungen erfüllt sind:

- Sie beginnt mit *file:* gefolgt von normalen Schrägstrichen:
  - drei Schrägstriche "///" für Dateien, die sich auf demselben Computer befinden wie der Browser (alternativ "//localhost/") oder
  - zwei Schrägstriche gefolgt vom Servernamen und einem weiteren Schrägstrich für Dateien auf Dateiservern, die vom Rechner, auf dem der Webbrowser läuft, erreichbar sind.
- Danach folgt der vollständige Dateipfad, der mit dem Dateinamen endet.
- Auf Microsoft Windows-Systemen wird der Pfad ebenfalls mit normalen Schrägstrichen anstelle von umgekehrten Schrägstrichen geschrieben.
- Der Laufwerksbuchstabe eines lokalen Pfads auf Microsoft Windows-Systemen wird wie üblich verwendet, zum Beispiel *C:*.
- Pfade mit Leerzeichen und Sonderzeichen wie "{, }, ^, #, ?" müssen auf Microsoft Windows-Systemen mit Prozentzeichen kodiert werden (z. B. mit "%20" für ein Leerzeichen).

Beispiel-URLs:

- file://file-server/path/to/my/file.ext
- file:///linux/local/file.pdf
- file:///C:/Users/myuser/localfile.doc

... ein **Label** definieren?

Wert für Annotation *text-type*: **label**

Dies ist ein schreibgeschütztes Feld, das grau angezeigt wird. Verwenden Sie die Annotation *label-group*, um das Label und die dazugehörigen Eingabefelder zu verknüpfen. Berücksichtigen Sie die Annotationen für Label (*show-label-in-edit*, *show-label-in-view*), bevor Sie spezielle Label-Felder implementieren!

... ein Feld für das **CM.Track-Login** definieren?

Wert für Annotation *username*: **true**

Wird für die Authentifizierung beim CM.Track-Server verwendet. Nur für Datenobjektgruppenfelder in einem Kontaktobjekt.

... ein Feld für das **CM.Track-Passwort** definieren?

Wert für Annotation *password*: **true**

Wird (im Datenbankmodus) für die Authentifizierung beim CM.Track-Server verwendet. Nur für Datenobjektgruppenfelder in einem Kontaktobjekt.

… ein Feld für **gültige E-Mail-Adressen** definieren?

Wert für Annotation *email*: **true**

Das Feld darf nur gültige E-Mail-Adressen enthalten. Die Eingabe wird gemäß des Standardformats für E-Mail-Adressen validiert <name>@<domain>.

… eine skriptbasierte Autocomplete-Liste definieren?

Wert für die Annotation *text-type* = **autocomplete**

Optional: Wert für die Annotation *autocomplete-script* = <Name des entsprechenden Skripts>

Eine skriptbasierte Autocomplete-Liste wird verwendet, um ein Drop-down-Menü bereitzustellen, das anhand der Eingabe, die der Bearbeiter bereits vorgenommen hat, dynamisch gefüllt wird. Wenn der Benutzer zum Beispiel "Mei" eingibt, werden die möglichen Werte "Meier", "Meister" und "Meinert" als Liste angezeigt und der Bearbeiter kann den für das Feld erforderliche Wert auswählen. Sie kennen dieses Verhalten von anderen Autocomplete-Feldern, z. B. der Suche nach Bearbeitern für ein Ticket oder die Suche nach Kunden bei der Ticketerstellung. In diesen Fällen erzeugt CM die Liste allerdings automatisch und das Verhalten kann nicht beeinflusst oder angepasst werden. Im Gegensatz dazu können skriptbasierte Autocomplete-Listen vom CM-Administrator implementiert werden. Die Werte basieren auf einem Ergebnissatz, der dynamisch erstellt wird. Der Ergebnissatu kann Strings, Bearbeiter, Kunden (Units) und Ressourcen enthalten.

Eine detaillierte Beschreibung von skriptbasierten Autocomplete-Listen finden Sie im Abschnitt *Skriptbasierte Autocomplete-Listen* des *Administratorhandbuchs*.

D.8.4 Benutzerdefinierte Felder für Ticketdaten

Im Admin Tool werden die Benutzerdefinierten Felder für Ticketdaten in der *Verwaltung der Benutzerdefinierten Felder* definiert: Navigationsgruppe *Tickets*, Navigationselement *Benutzerdefinierte Felder*.

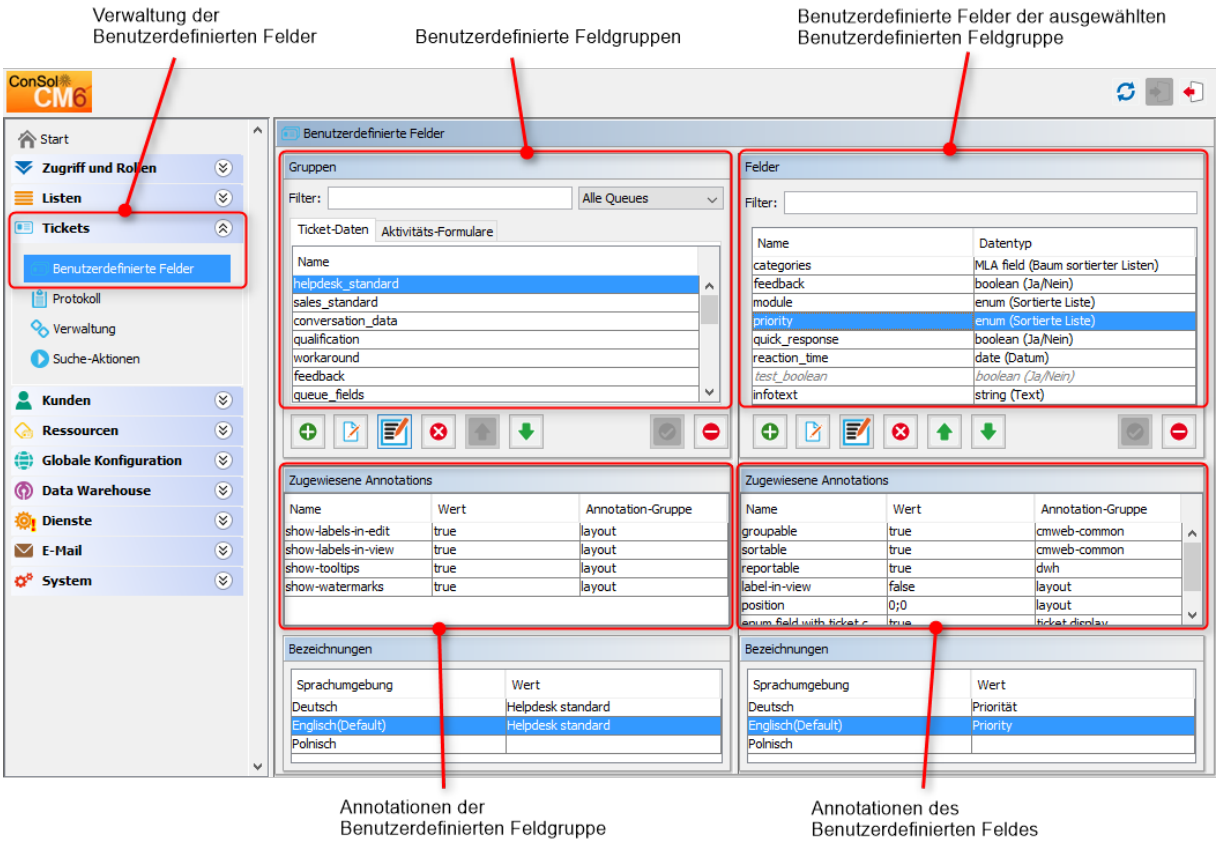


Abbildung 122: *ConSol CM Admin Tool: Verwaltung der Benutzerdefinierten Felder für Ticketdaten (CM-Version 6.10)*

D.8.4.1 Die wichtigsten Methoden für den Zugriff auf Benutzerdefinierte Felder für Tickets

Die folgenden drei Methoden sind für die Programmierung des Zugriffs auf Benutzerdefinierte Felder in CM-Skripten wichtig. Alle Methoden gehören zur Klasse *Ticket*:

- **Ticket.get()**
  - Zum Abrufen von Daten aus einem Benutzerdefinierten Feld.
- **Ticket.set()**
  - Zum Setzen von Daten in einem bereits vorhandenen Benutzerdefinierten Feld.
- **Ticket.add()**
  - Für Berechnungen mit einem Wert in einem Benutzerdefinierten Feld, z. B. zum Hinzufügen eines bestimmten Zeitraums zu einem Feld des Datentyps *date*.
  - Zum Hinzufügen einer neuen Zeile in Listenfeldern (einfache Listen und Tabellen).

Eine weitere Methode kann verwendet werden, wenn ein Feld geleert werden soll, d. h. wenn der Wert auf *null* gesetzt werden soll:

- **Ticket.remove()**
  - Setzt den Wert des Felds auf *null*.

D.8.4.2 Abrufen von Werten von Benutzerdefinierten Feldern für Ticketdaten

Um Daten aus einem Benutzerdefinierten Feld in einem Skript abzurufen, müssen Sie es über die technischen Namen der Benutzerdefinierten Feldgruppe und des Benutzerdefinierten Feldes referenzieren. Die zu verwendende Methode kann je nach Datentyp des Benutzerdefinierten Feldes unterschiedlich sein.

Einfache Datentypen

Die folgenden Beispiele beziehen sich auf die Benutzerdefinierten Felder aus der obigen Abbildung. Die zu verwendende Methode (der bequemste Weg) ist:

```
ticket.get("<Gruppenname>.<Feldname>")
```

Denken Sie daran, dass die *Getter-Methode* für ein Feld abhängig vom Typ des Datenfelds entweder einen Wert oder ein Objekt zurückgeben kann! Dies ist in folgendem Beispiel erklärt.

Mit dem folgenden Admin-Tool-Skript, das z. B. aus dem Workflow aufgerufen wird, werden Ticketdaten angezeigt:

```
// display info from custom types of various data types:
import com.consol.cmas.common.model.ticket.Ticket

Ticket ticket = workflowApi.ticket
println 'Display enum from helpdesk CF group ... '

def myfield = ticket.get("helpdesk_standard.priority")
println 'Class of helpdesk_standard.priority field is ' + myfield.getClass()
println 'Value of helpdesk_standard.priority field is ' + myfield.getName()
println 'Retrieve value directly, method 1, using getter method ... '

def my_new_field1 = ticket.get("helpdesk_standard.priority").getName()
println 'Result is ' + my_new_field1
println 'Retrieve value directly, method 2, using direct access to attribute ... '

def my_new_field2 = ticket.get("helpdesk_standard.priority").name
println 'Result is ' + my_new_field2
println 'Display value of simple data field ...'

def fb = ticket.get("helpdesk_standard.feedback")
println 'Value of feedback boolean field is ' + fb
```

Code-Beispiel 25: *Admin-Tool-Skript zum Anzeigen von Benutzerdefinierten Feldern*

Die folgende Ausgabe wird in der Datei server.log angezeigt:

```
Display enum from helpdesk CF group ...
Class of helpdesk_standard.priority field is class com.consol.cmas.common.model.customfield.enums.EnumValue_$$_jvstb5d_61
Value of helpdesk_standard.priority field is low
Retrieve value directly, method 1, using getter method ...
Result is low
Retrieve value directly, method 2, using direct access to attribute ...
Result is low
 Display value of simple data field ...
Value of feedback boolean field is true
```

Erklärung:

Wenn Sie den Wert eines Benutzerdefinierten Feldes, das ein Objekt enthält, (z. B. einen EnumValue) abrufen möchten, müssen Sie Methoden wie *getName()* verwenden, um den tatsächlichen Wert abzurufen, da ticket.get ... nur das Objekt liefert. Die Notation *.name* ist eine vereinfachte Version (Groovy) der Getter-Methode.

Wenn Sie den Wert eines einfachen Datenfeldes abrufen möchten, können Sie dies "direkt" tun: ticket.get ... liefert den Wert.

Ein Bedingungsskript einer Workflow-Aktivität könnte wie folgender Code aussehen:

```
boolean vip_info = ticket.get("am_fields.vip")

if(vip_info == true){
   return true
}
else {
   return false
}
```

Code-Beispiel 26: *Bedingungsskript, in dem ein Boolean-Wert überprüft wird*

Oder kürzer:

```
return ticket.get("am_fields.vip")
```

Code-Beispiel 27: *Bedingungsskript, in dem ein Boolean-Wert überprüft wird, Kurzversion*

Listenwerte

Ein Feld des Typs *enum (sortierte Liste)* ist ein Feld, in dem der Wert einer von mehreren Listenwerten ist. Zum Beispiel kann eine Liste mit Prioritäten die Grundlage für ein Enum-Feld bilden. Sie können zum Abrufen des Wertes eines Enum-Feldes die gleiche Syntax verwenden, wie für einfache Datentypen. Die Methode *get* liefert den Listenwert des Enums und die Methode *getName()* liefert den Namen des Wertes als *String-Attribut*.

```
def prio = ticket.get("helpdesk_standard.priority")
println "Priority is now " + prio.getName()
//or shorter:
println "Priority is now " + prio.name
```

Code-Beispiel 28: *Abrufen eines Enum-Wertes für ein Benutzerdefiniertes Feld*

MLAs

Arbeiten mit einem MLA-Wert (das ausgewählte "Blatt" des MLA-Baums)

Die Java-Klasse für MLAs (Multi Level Attributes) ist *MultiEnumField*. Sie wird wie ein normales Enum-Feld behandelt, d. h. Sie können Java-/Groovy-Code wie im nachfolgenden Beispiel verwenden, um den aktuellen Wert eines MLA-Feldes abzurufen.

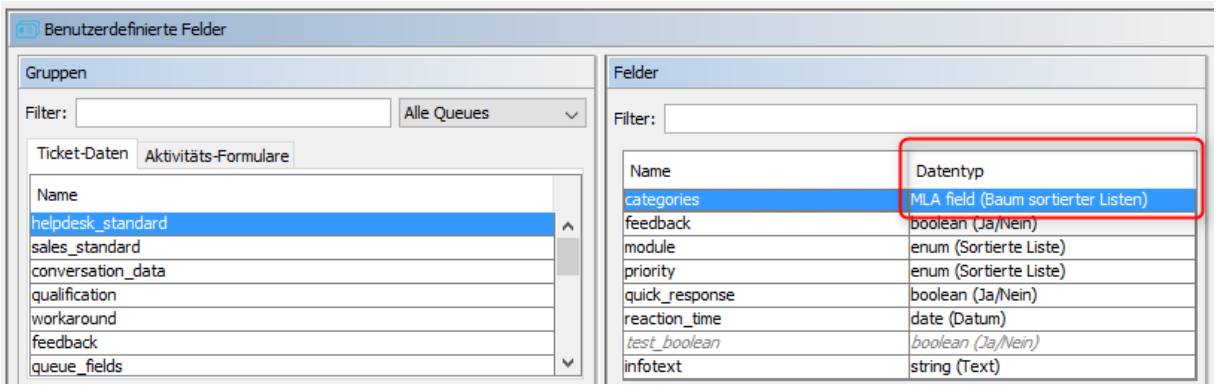


Abbildung 123: *Admin Tool: Definition eines Benutzerdefinierten Feldes des Typs MLA*

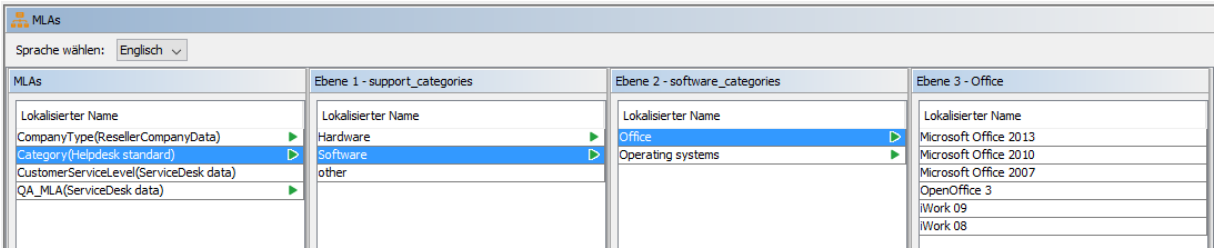


Abbildung 124: *Admin Tool: MLA-Definition*

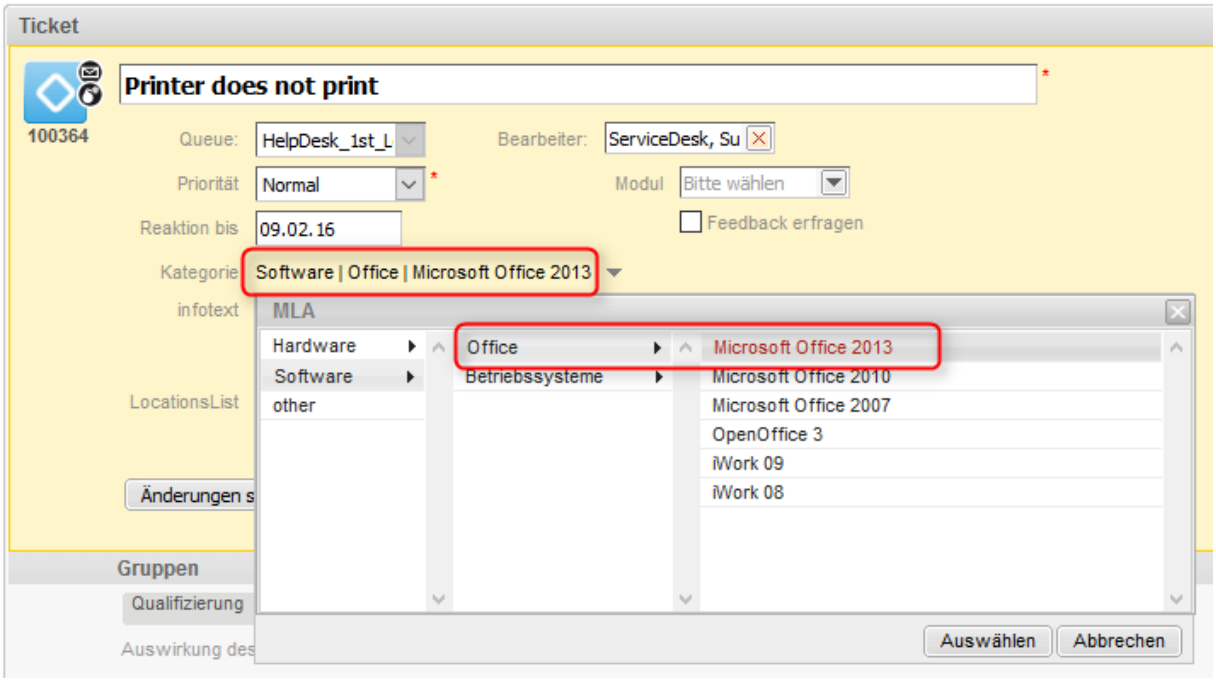


Abbildung 125: *Web Client: Setzen eines Wertes (technisch: Name) eines MLAs*

```
// Get
def myEnumValueName = ticket.get("GROUP_NAME.MLA_FIELD_NAME").name
// Set
EnumValue enumValue = enumService.getValueByName("ENUM_NAME","ENUM_VALUE_NAME")
ticket.set("GROUP_NAME.MLA_FIELD_NAME", enumValue)
```

Beispiel mit technischem Namen des Feldes und lokalisiertem Namen:

```
import com.consol.cmas.common.model.customfield.enums.EnumValue

println 'Displaying MLA value ...'
def my_mla_field = ticket.get("helpdesk_standard.categories")
println 'MLA value categories (technical name) is now ' + my_mla_field.name
def my_mla_field_localized = localizationService.getLocalizedProperty
  (EnumValue.class, "name", my_mla_field.id, engineerService.getCurrentLocale())
println 'MLA value categories (localized name) is now ' + my_mla_field_localized
```

Code-Beispiel 29: *Code (Workflow- oder Admin-Tool-Skript) für die Anzeige der MLA-Daten*

```
stdout] [Susan-] Displaying MLA value ...
stdout] [Susan-] MLA value categories (technical name) is now ms_2013
stdout] [Susan-] MLA value categories (localized name) is now Microsoft Office 2013
```

Abbildung 126: *Log-Ausgabe für das obige Beispiel*

Arbeiten mit dem gesamten Pfad des ausgewählten MLA-Wertes

Wenn Sie den gesamten Pfad des ausgewählten MLA-Wertes abrufen möchten, können Sie folgendermaßen vorgehen:

```
println 'Displaying MLA branch ...'
List<EnumValue> mlaPathElements = mlaService.getAssignedMla(ticket, "helpdesk_
 standard", "categories")
def mlaPath1=""
mlaPathElements.each() {elem ->
   mlaPath1 = elem.name + ' -- ' + mlaPath1
}
println 'mlaPath1 is now ' + mlaPath1

//or shorter:
def mlaPath2 = mlaService.getAssignedMla(ticket, "helpdesk_standard",
 "categories").reverse()*.name.join(" -- ")
println 'mlaPath2 is now ' + mlaPath2
```

Code-Beispiel 30: *Abrufen des gesamten Pfads zum ausgewählten MLA-Wert*

```
stdout] [Susan-] Displaying MLA branch ...
stdout] [Susan-] mlaPath1 is now software -- office -- ms_2013 --
stdout] [Susan-] mlaPath2 is now software -- office -- ms_2013
```

Abbildung 127: *Log-Ausgabe für das obige Beispiel*

Listen

Listen mit einfachen Datentypen

Eine Liste mit einfachen Datentypen besteht aus einer Liste (= Array), bei der in jeder Zeile ein Wert eines einfachen Datentyps steht, in unserem Beispiel ein *Datum*. Das Benutzerdefinierte Feld des Typs *date* muss den Parameter *Gehört zu* haben, der auf die Liste zeigt.

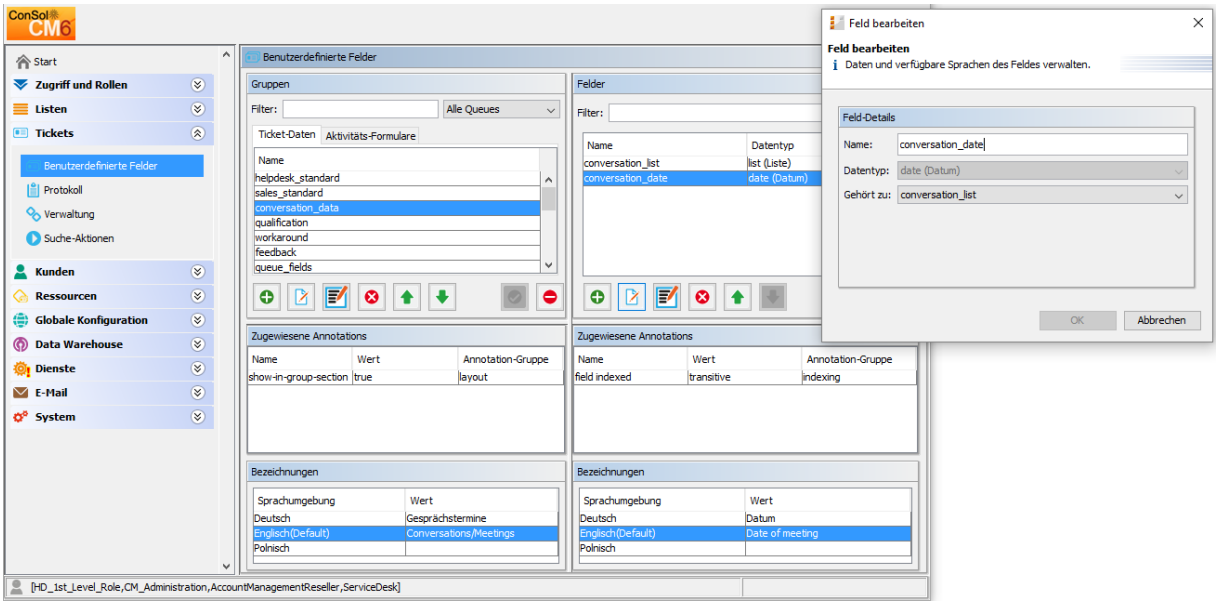


Abbildung 128: *ConSol CM Admin Tool - Benutzerdefiniertes Feld für eine Liste mit Datumsfeldern*

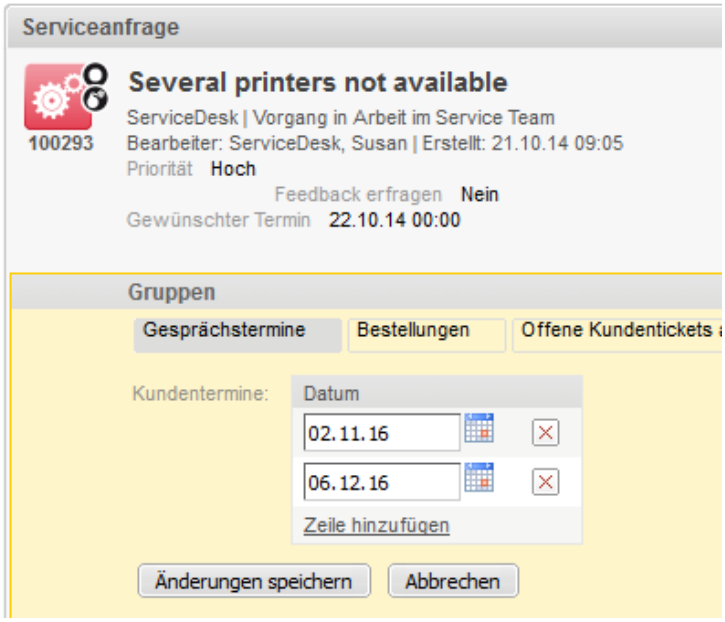


Abbildung 129: *ConSol CM Web Client - Liste der Datumsfelder in einem Ticket (Bearbeitungsmodus)*

Mit folgendem Code können Sie auf die einzelnen Benutzerdefinierten Felder des Typs *date* der Liste zugreifen:

```
def convs = ticket.get("conversation_data.conversation_list").each() { conv ->
   println "NEXT DATE is :" + conv
   println "CLASS of NEXT DATE is " + conv.getClass()
}
```

Code-Beispiel 31: *Anzeigen des Inhalts einer Liste mit Datenobjekten*

```
2014-03-27 12:21:45,274 INFO  [STDOUT                    ] NEXT DATE is :2014-02-03 00:00:00.0
2014-03-27 12:21:45,279 INFO  [STDOUT                    ] CLASS of MEXT DATE is class java.sql.Timestamp
2014-03-27 12:21:45,280 INFO  [STDOUT                    ] NEXT DATE is :2014-03-02 00:00:00.0
2014-03-27 12:21:45,280 INFO  [STDOUT                    ] CLASS of MEXT DATE is class java.sql.Timestamp
2014-03-27 12:21:45,281 INFO  [STDOUT                    ] NEXT DATE is :2014-03-26 00:00:00.0
2014-03-27 12:21:45,281 INFO  [STDOUT                    ] CLASS of MEXT DATE is class java.sql.Timestamp
```

Abbildung 130: *Log-Ausgabe des obigen Skripts*

Um auf eine bestimmte Zeile zuzugreifen, können Sie folgende Syntax verwenden:

```
def mydate = ticket.get("conversation_data.conversation_list[1]")
```

Code-Beispiel 32: *Abrufen eines bestimmten Werts aus einer Liste mit einfachen Datentypen*

Lists of Structs (Tabellen)

Die Datenstruktur *list of structs* bildet die technische Grundlage für eine Tabelle mit mehreren Spalten im Web Client. Die Liste ist das übergeordnete Objekt, das Zeilen enthält. Jede Zeile ist eine Instanz eines struct. Jede Zeile (struct) enthält so viele Benutzerdefinierte Felder (Tabellenspalten) wie erforderlich. Siehe Abbildung in der Einleitung dieses Abschnitts.

Um die Daten aus einer List of Structs abzurufen, arbeiten Sie mit einer Iteration über die Zeilen (= structs). Im folgenden Beispiel (aus einem Bestellsystem, nicht in der obigen Abbildung gezeigt), arbeiten wir mit einer Tabelle, in der ...

- das Benutzerdefinierte Feld *orders_list* die Liste darstellt.
- sich das Benutzerdefinierte Feld *orders_list* innerhalb der Benutzerdefinierten Feldgruppe *order_data* befindet.
- der Iterator *str* das struct darstellt.
- das struct drei Felder hat:
    - *orders_hardware*
      ist der Artikel, der bestellt werden soll (*enum*).
    - *orders_contact*
      ist die Kontaktperson (*string*).
    - *orders_number*
      ist die Anzahl der Artikel, die bestellt werden sollen (*integer*).

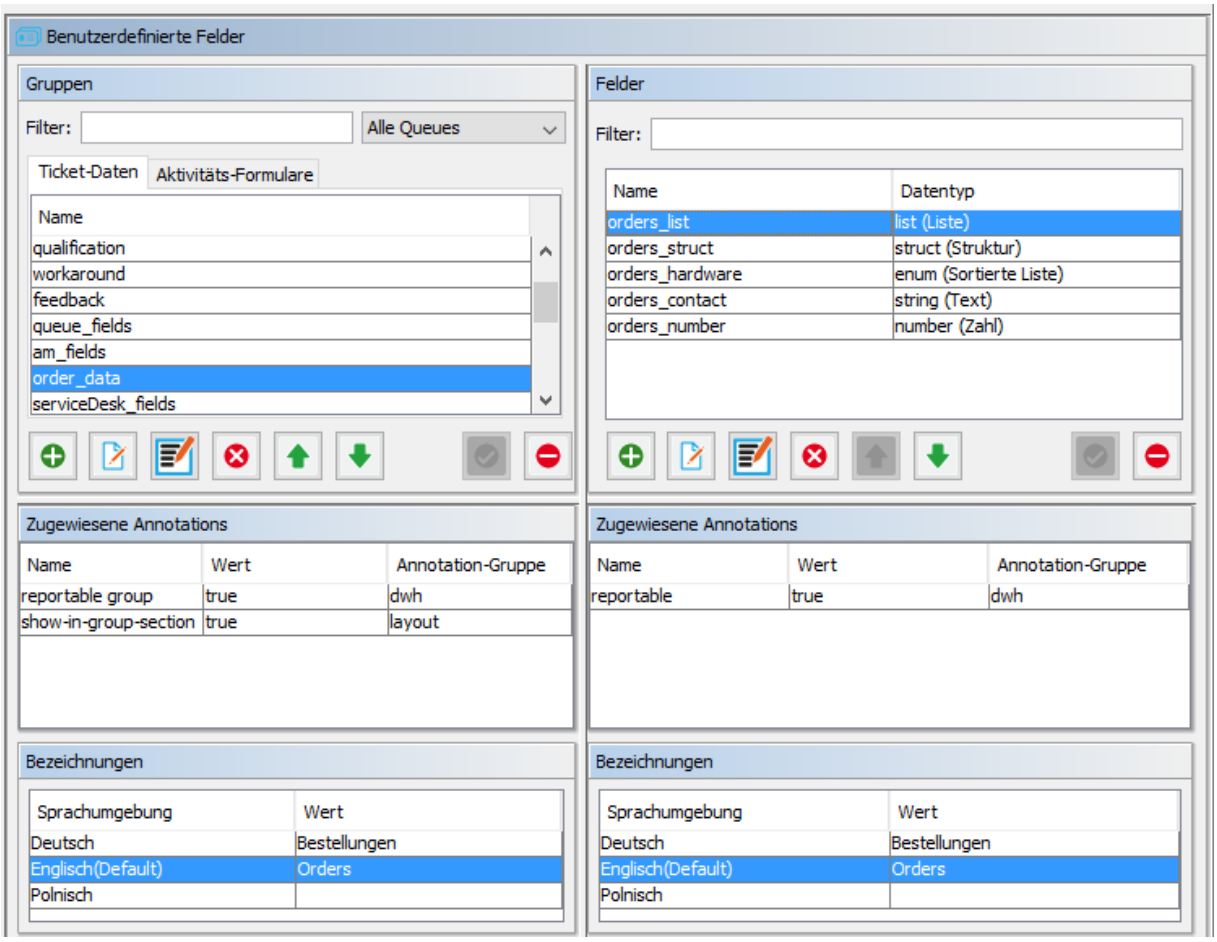


Abbildung 131: *ConSol CM Admin Tool - Benutzerdefinierte Felder für die Liste*

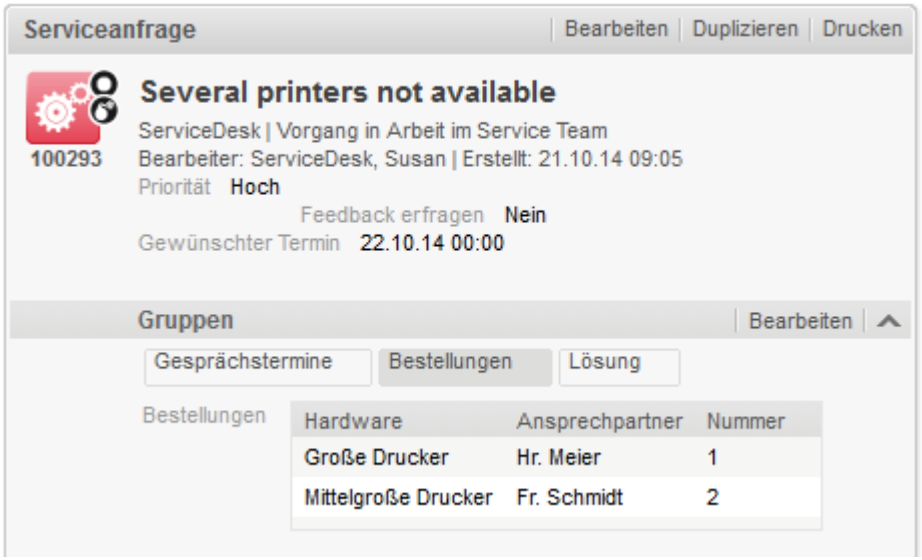


Abbildung 132: *ConSol CM Web Client - Ticket mit ausgefüllter Tabelle*

```
def structs = ticket.get("order_data.orders_list").each() { str ->
   println("CLASS of LINE is " + str.getClass())
   println("FIELD VALUE HARDWARE is " + str.orders_hardware.getName())
   println("CLASS of FIELD VALUE HARDWARE is " + str.orders_hardware.getName
     ().getClass())
   println("FIELD VALUE CONTACT is " + str.orders_contact)
   println("CLASS of FIELD VALUE CONTACT is " + str.orders_contact.getClass())
   println("FIELD VALUE NUMBER is " + str.orders_number)
   println("CLASS of FIELD VALUE NUMBER is " + str.orders_number.getClass())
}
```

Code-Beispiel 33: *Abrufen von Daten aus einer List of Structs*

```
2014-03-27 11:39:44,425 INFO  [STDOUT                    ] CLASS of LINE is class com.consol.cmas.common.model.customfield.cfel.Struct
2014-03-27 11:39:44,429 INFO  [STDOUT                    ] FIELD VALUE HARDWARE is large_printers
2014-03-27 11:39:44,429 INFO  [STDOUT                    ] CLASS of FIELD VALUE HARDWARE is class java.lang.String
2014-03-27 11:39:44,430 INFO  [STDOUT                    ] FIELD VALUE CONTACTis Mr. Miller
2014-03-27 11:39:44,431 INFO  [STDOUT                    ] CLASS of FIELD VALUE CONTACT is class java.lang.String
2014-03-27 11:39:44,431 INFO  [STDOUT                    ] FIELD VALUE NUMBER is 2
2014-03-27 11:39:44,454 INFO  [STDOUT                    ] CLASS of FIELD VALUE NUMBER is class java.lang.Long
2014-03-27 11:39:44,455 INFO  [STDOUT                    ] CLASS of LINE is class com.consol.cmas.common.model.customfield.cfel.Struct
2014-03-27 11:39:44,456 INFO  [STDOUT                    ] FIELD VALUE HARDWARE is medium_printers
2014-03-27 11:39:44,456 INFO  [STDOUT                    ] CLASS of FIELD VALUE HARDWARE is class java.lang.String
2014-03-27 11:39:44,456 INFO  [STDOUT                    ] FIELD VALUE CONTACTis Mrs. Summer
2014-03-27 11:39:44,457 INFO  [STDOUT                    ] CLASS of FIELD VALUE CONTACT is class java.lang.String
2014-03-27 11:39:44,458 INFO  [STDOUT                    ] FIELD VALUE NUMBER is 5
2014-03-27 11:39:44,458 INFO  [STDOUT                    ] CLASS of FIELD VALUE NUMBER is class java.lang.Long
```

Abbildung 133: *Log-Datei - Skriptausgabe*

D.8.4.3 Setzen von Werten von Benutzerdefinierten Feldern für Ticketdaten

Um Werte für Benutzerdefinierte Felder für Tickets zu setzen, folgende Sie dem gleichen Prinzip wie für den Abruf der Daten: Verwenden Sie den Namen der Benutzerdefinierten Feldgruppe und den technischen Namen des Benutzerdefinierten Feldes als Referenz. Außerdem ist natürlich der neue Wert erforderlich. Dieser muss selbstverständlich den richtigen Datentyp haben.

```
ticket.set("<Gruppenname>.<Feldname>", <Wert>)
```

Setzen von Werten für Benutzerdefinierte Felder mit einfachen Datentypen

```
ticket.set("fields.reaction_time", new Date());
```

Code-Beispiel 34: *Setzen eines Wertes für ein Benutzerdefiniertes Feld mit einem Datum*

Wenn Sie mit Feldern des Datentyps *number* oder *date* arbeiten, können Sie die Werte der Benutzerdefinierten Felder bequem berechnen, siehe folgendes Beispiel.

```
//add 24 hours (in millis) to current field value
ticket.add("fields.deadline", 24*60*60*1000)
```

Code-Beispiel 35: *Berechnen des Wertes eines Benutzerdefinierten Feldes mit einem Datum*

Das Setzen eines Wertes auf *null* (d. h. Leeren des Feldes) ist dasselbe wie das Entfernen des Wertes:

```
ticket.set("fields.numberOfEmployees", null)
```

Code-Beispiel 36: *Setzen des Wertes eines Benutzerdefinierten Feldes auf null*

Oder kürzer:

```
ticket.remove("fields.numberOfEmployees" )
```

Code-Beispiel 37: *Setzen des Wertes eines Benutzerdefinierten Feldes auf null durch Entfernen des Wertes*

Setzen von Enum-Werten

Verwenden Sie folgende Syntax, um einen *enum-Wert* zu setzen. Der neue Wert muss natürlich in der Sortierten Liste (enum) vorhanden sein, die vom Benutzerdefinierten Feld referenziert wird.

```
ticket.set("Gruppenname.Feldname",<technischer Name des Wertes>)
```

```
ticket.set("fields.priority", "URGENT");
```

Code-Beispiel 38: *Setzen eines Enum-Wertes*

Setzen von Listenwerten

Setzen von Werten in Listen mit einfachen Datentypen

Wenn Sie eine Zeile hinzufügen möchten, können Sie einfach die Methode *add* verwenden:

```
ticket.add("fields.tags", "my new String")
```

Code-Beispiel 39: *Hinzufügen einer neuen Zeile zu einer Liste mit Strings*

Wenn Sie sich auf einen bestimmten Wert beziehen möchten, um einen neuen Wert dafür zu setzen, müssen Sie die Syntax für ein Array verwenden:

```
ticket.set("fields.tags[last]", "ConSol CM6")
```

Code-Beispiel 40: *Setzen eines Werts in einer Liste mit Strings*

Setzen von Werten in einer List of Structs

Wenn Sie mit *structs* arbeiten, müssen Sie immer mit dem Schlüssel des Wertes arbeiten, den Sie hinzufügen oder setzen möchten. Wenn Sie eine neue Zeile *hinzufügen* möchten, müssen Sie ein neues Struct als neue Zeile erstellen. Mit der Methode *set* können Sie nacheinander jedes neue Feld befüllen.

```
ticket.add("order_data.orders_list", new Struct()
.set("tA_Id", id).set("orders_hardware",mynewhardware_model)
.set("orders_contact", thenewcontactname)
.set("orders_number",thenewnumber)
```

Code-Beispiel 41: *Hinzufügen einer neuen Zeile zu einer List of Structs*

D.8.4.4 Ein- und Ausblenden von Benutzerdefinierten Feldgruppen

Eine Benutzerdefinierte Feldgruppe kann mit einer Methode von *workflowApi* eingeblendet (sichtbar gemacht) und ausgeblendet (unsichtbar gemacht) werden. Dies funktioniert für Benutzerdefinierte Feldgruppen, die im Kopfbereich des Tickets angezeigt werden, und für Benutzerdefinierte Feldgruppen, die im Gruppenbereich des Tickets angezeigt werden.

Ein typischer Anwendungsfall ist eine Benutzerdefinierte Feldgruppe, die zuerst unsichtbar ist (Gruppenannotation *group-visibility = false*) und zu dem Zeitpunkt eingeblendet wird, an dem der Bearbeiter im Prozess mit den Daten arbeiten muss. Eine Benutzerdefinierte Feldgruppe, die die Gründe für die Ablehnung einer Anfrage enthält, wird zum Beispiel erst angezeigt (eingeblendet), wenn der Bearbeiter auf die Workflow-Aktivität *Ticket verwerfen ...* geklickt hat. Dies verhindert, dass das Ticket mit Informationen überladen wird.

```
workflowApi.setGroupProperty("CF_Group_Dismissal",GroupPropertyType.VISIBLE,"true")
```

Code-Beispiel 42: *Einblenden einer Benutzerdefinierten Feldgruppe*

Um einige Benutzerdefinierte Feldgruppen auszublenden, z. B. wenn das Ticket qualifiziert wurde und einige der Benutzerdefinierten Feldgruppen für den Prozess nicht mehr benötigt werden, können Sie Code wie den im folgenden Beispiel gezeigten verwenden:

```
workflowApi.setGroupProperty("CF_Group_
 HardwareInfo",GroupPropertyType.VISIBLE,"false")
```

```
workflowApi.setGroupProperty("CF_Group_
 SoftwareInfo",GroupPropertyType.VISIBLE,"false")
```

Code-Beispiel 43: *Ausblenden von Benutzerdefinierten Feldgruppen*

D.8.5 Datenfelder für Kundendaten

D.8.5.1 Datenobjektgruppenfelder für Kundendaten (CM-Version 6.9 und höher)

In CM-Version 6.9 und höher gehören Datenobjektgruppen für Kunden zum neuen Kundendatenmodell (*FlexCDM*). In Version 6.9 werden sie im Admin Tool, Abschnitt *Benutzer-Attribute*, Tab *Kundendatenmodell* definiert. In Version 6.10 werden sie im Admin Tool, Navigationsgruppe *Kunden*, Navigationselement *Datenmodelle* definiert (siehe auch Datenobjektgruppenfelder für Kundendaten (CM-Version 6.10 und höher).

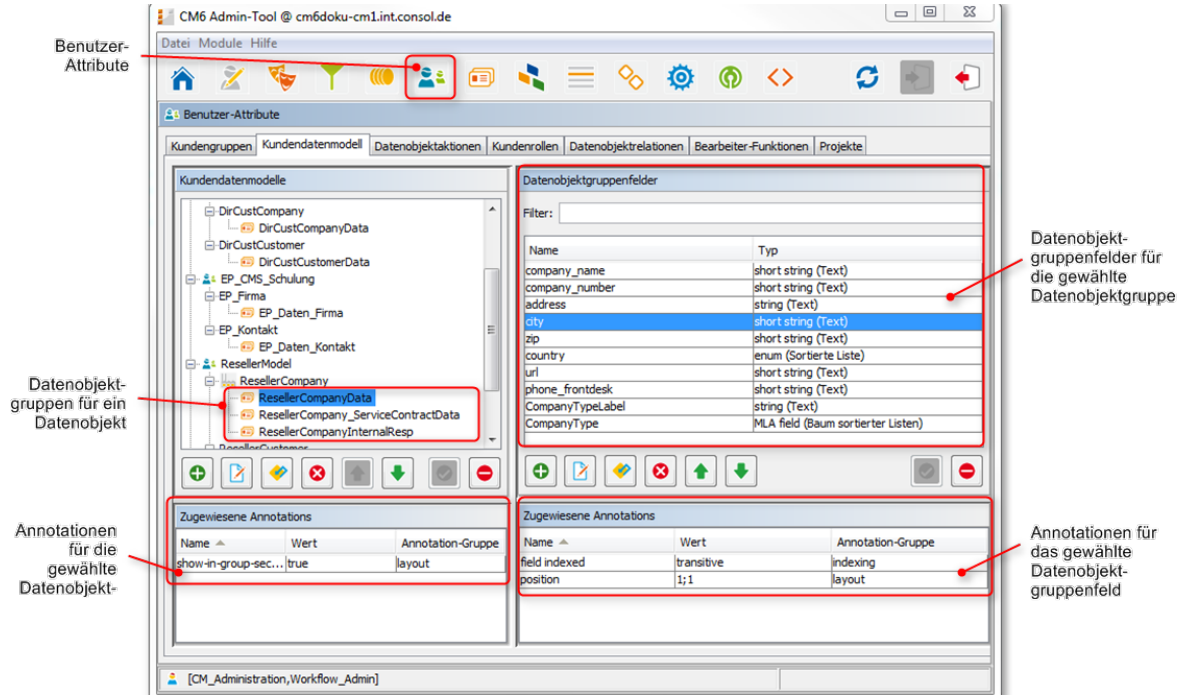


Abbildung 134: *ConSol CM Admin Tool - Verwaltung von Datenobjektgruppenfeldern für Kundendaten (CM-Version 6.9)*

Die Felder, die in Kundendatenmodellen früherer Versionen Benutzerdefinierte Felder hießen, heißen jetzt **Datenobjektgruppenfelder**. Das Prinzip, über das Sie Werte für diese Datenfelder abrufen und setzen, ist allerdings grundsätzlich das gleiche wie in CM-Version 6.8 und älter.

Die wichtigsten Methoden für den Zugriff auf Datenobjektgruppenfelder für Kundendaten

Die folgenden drei Methoden sind für die Programmierung des Zugriffs auf Datenobjektgruppenfelder in CM-Skripten wichtig. Alle Methoden gehören zur Klasse *Unit*:

- **Unit.get()**
  - Zum Abrufen von Daten aus einem Datenobjektgruppenfeld.
- **Unit.set()**
  - Zum Setzen von Daten in einem bereits vorhandenen Datenobjektgruppenfeld.

- **Unit.add()**
  - Für Berechnungen mit einem Wert in einem Datenobjektgruppenfeld, z. B. zum Hinzufügen eines bestimmten Zeitraums zu einem Feld des Datentyps *date*.
  - Zum Hinzufügen einer neuen Zeile in Listenfeldern (einfache Listen und Tabellen).

Eine andere Methode kann verwendet werden, wenn ein Feld geleert werden soll, d. h. wenn der Wert auf *null* gesetzt werden soll:

- **Unit.remove()**
  - Setzt den Wert des Felds auf *null*.

Abrufen von Werten für Kundendaten in CM-Version 6.9 und höher

Da der Name eines *Datenobjektgruppenfeldes* in mehr als einer *Datenobjektgruppe* vorkommen kann, muss der Name der Datenobjektgruppe beim Zugriff auf die Kundendaten angegeben werden. Zum Beispiel könnten im in der obigen Abbildung gezeigten Kundendatenmodell die Datenobjektgruppen *ResellerCompanyData* und *DirCustCompanyData* ein Datenobjektgruppenfeld mit dem Namen *city* haben. Deshalb ist es wichtig, den Gruppennamen und den Feldnamen anzugeben.

Verwenden Sie dazu folgende Syntax:

```
unit.get("group1:name")
```

Zum Beispiel:

```
def mycity = company.get("ResellerCompanyData:city")
```

Code-Beispiel 44: *Abrufen eines Feldwertes für eine Firma*

Es gibt mehrere Objekte und Methoden, mit denen Sie auf unterschiedlichen Ebenen von FlexCDM mit Daten arbeiten können. Im folgenden Beispiel werden mehrere häufige Objekte und Methoden angewendet. Dies ist ein Admin-Tool-Skript, auf das aus einer Workflow-Aktivität zugegriffen wird. Der einzige Zweck ist die Anzeige einiger Daten des Hauptkunden des Tickets. Die folgende Abbildung zeigt die im Skript verwendeten Java-Objekte und die referenzierten ConSol CM-Objekte im Admin Tool.

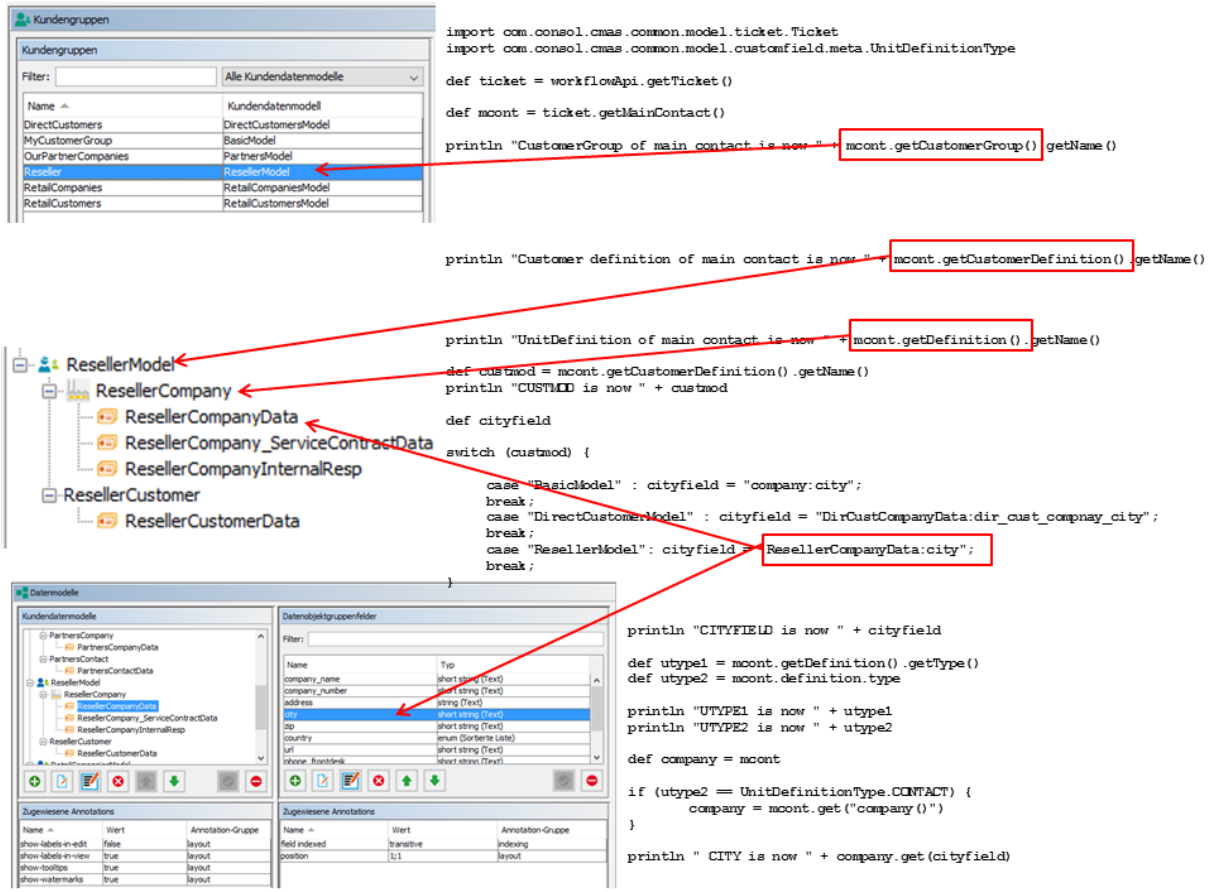
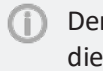


Abbildung 135: *ConSol CM-Kundenobjekte im Skript und Admin Tool*



Denken Sie daran, dass Sie für Getter-Methoden wie *unit.getDefinition().getType()* auch die kurze Notation wie *unit.definition.type* verwenden können.

```
import com.consol.cmas.common.model.ticket.Ticket
import com.consol.cmas.common.model.customfield.meta.UnitDefinitionType

def ticket = workflowApi.getTicket()

def mcont = ticket.getMainContact()
println "CustomerGroup of main contact is now " + mcont.getCustomerGroup().getName
  ()
println "Customer definition of main contact is now " + mcont.getCustomerDefinition
  ().getName()
println "UnitDefinition of main contact is now " + mcont.getDefinition().getName()

def custmod = mcont.getCustomerDefinition().getName()
// println "CUSTMOD is now " + custmod

def cityfield

switch (custmod) {
   case "BasicModel" : cityfield = "company:city";
   break;
   case "DirectCustomerModel" : cityfield = "DirCustCompanyData:dir_cust_company_
    city";
   break;
   case "ResellerModel": cityfield = "ResellerCompanyData:city";
   break;
}


println "CITYFIELD is now " + cityfield

def utype1 = mcont.getDefinition().getType()
def utype2 = mcont.definition.type

println "UTYPE1 is now " + utype1
println "UTYPE2 is now " + utype2

def company = mcont

if (utype2 == UnitDefinitionType.CONTACT) {
   company = mcont.get("company()")
}

def mycity = company.get(cityfield)
println " CITY is now " + mycity
```

Code-Beispiel 45: *Admin-Tool-Skript (aus dem Workflow aufgerufen) zum Anzeigen von Kundendaten*

Die Ausgabe der Log-Datei für den folgenden Datensatz ist unten abgebildet. Es wird das Modell *Reseller* aus der obigen Abbildung verwendet.

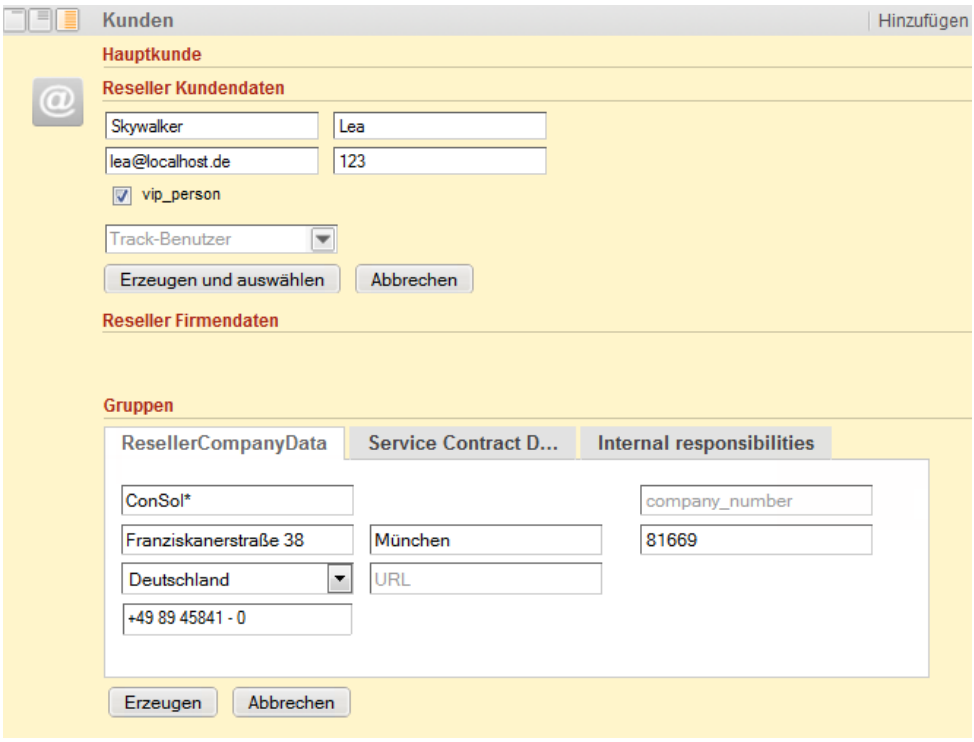


Abbildung 136: *ConSol CM Web Client - Kundendatensatz*

```
2014-03-27 16:09:21,739 INFO  [STDOUT                    ] CustomerGroup of main contact is now Reseller
2014-03-27 16:09:21,739 INFO  [STDOUT                    ] Customer definition of main contact is now ResellerModel
2014-03-27 16:09:21,740 INFO  [STDOUT                    ] UnitDefinition of main contact is now ResellerCustomer
2014-03-27 16:09:21,743 INFO  [STDOUT                    ] CUSTMOD is now ResellerModel
2014-03-27 16:09:21,744 INFO  [STDOUT                    ] CITYFIELD is now ResellerCompanyData:city
2014-03-27 16:09:21,750 INFO  [STDOUT                    ] UTYPE1 is now CONTACT
2014-03-27 16:09:21,751 INFO  [STDOUT                    ] UTYPE2 is now CONTACT
2014-03-27 16:09:21,759 INFO  [STDOUT                    ]  CITY is now München
```

Abbildung 137: *Log-Datei - Skriptausgabe für FlexCDM*

```
String firstName = company.get("responsibleConsultants[0].firstName");
```

Code-Beispiel 46: *Abrufen eines Wertes aus einer List of Structs mithilfe der Index-Notation*

Setzen von Werten für Kundendaten in CM-Version 6.9 und höher

Setzen von Werten für Datenobjektgruppenfelder mit einfachen Datentypen

Die Methoden *set* und *add* funktionieren so, wie es für Benutzerdefinierte Felder für Tickets beschrieben ist. Zum Beispiel:

```
//set number field
company.set("numberOfEmployees", 1);
//add 1 to field value, afterwards the value of the field is 2
company.add("numberOfEmployees", 1);
```

Code-Beispiel 47: *Setzen und Hinzufügen von Werten für ein Datenobjektgruppenfeld des Typs integer*

Listen

Setzen von Werten in einer List of Structs für Kundendaten

```
company.set("responsibleConsultants", [
   new Struct().set("lastName", "Miller").set("email", "miller@consol.com"),
   new Struct().set("lastName", "Smith").set("email", "smith@consol.com"),
   new Struct().set("lastName", "Burger").set("email", "burger@consol.com")
]);
```

Code-Beispiel 48: *Erstellen einer neuen List of Structs, Version 2*

```
company.add("responsibleConsultants", new Struct().set("lastName", " Nowitzki
 ").set("email", "dnowitzki@consol.us"));
```

Code-Beispiel 49: *Hinzufügen einer neuen List of Structs für Kundendaten*

```
company.set("responsibleConsultants[0].firstName", "John");
```

Code-Beispiel 50: *Setzen eines Wertes in einer List of Structs mithilfe der Index-Notation*

```
company.set("responsibleConsultants[last]", null);
```

Code-Beispiel 51: *Entfernen eines struct (= Zeile) aus einer List of Structs (= Tabelle)*

D.8.5.2 Convenience-Methoden für den Zugriff auf Kundendaten in CM-Version 6.9 und höher

```
Unit mainContact = ticket.getMainContact();

// "company" extension returns company for contact
Unit company = mainContact.get("company()");

// it is also possible to set company using "company" extension
mainContact.set("company()", company);

// "contacts" extension returns list of contacts for company
List contacts = company.get("contacts()");

// "tickets" extension returns list of tickets for contact or company
List tickets = company.get("tickets()");
tickets = mainContact.get("tickets()");

// extensions can be chained
Integer count = contact.get("company().contacts()[0].tickets()[count]");

// parentheses can be omitted, but it is not recommended (possible collision with
 name of group or field)
count = contact.get("company.contacts[0].tickets[count]"); // here "company" is not
 extension but name of field
```

Code-Beispiel 52: *Convenience-Methoden für den Zugriff auf Kundendaten*

D.8.5.3 Datenobjektgruppenfelder für Kundendaten (CM-Version 6.10 und höher)

In CM-Version 6.10 und höher gehören Datenobjektgruppen für Kundendaten, wie in Version 6.9, zum Kundendatenmodell (*FlexCDM*). Sie werden im Admin Tool, Navigationsgruppe *Kunden*, Navigationselement *Datenmodelle* definiert.

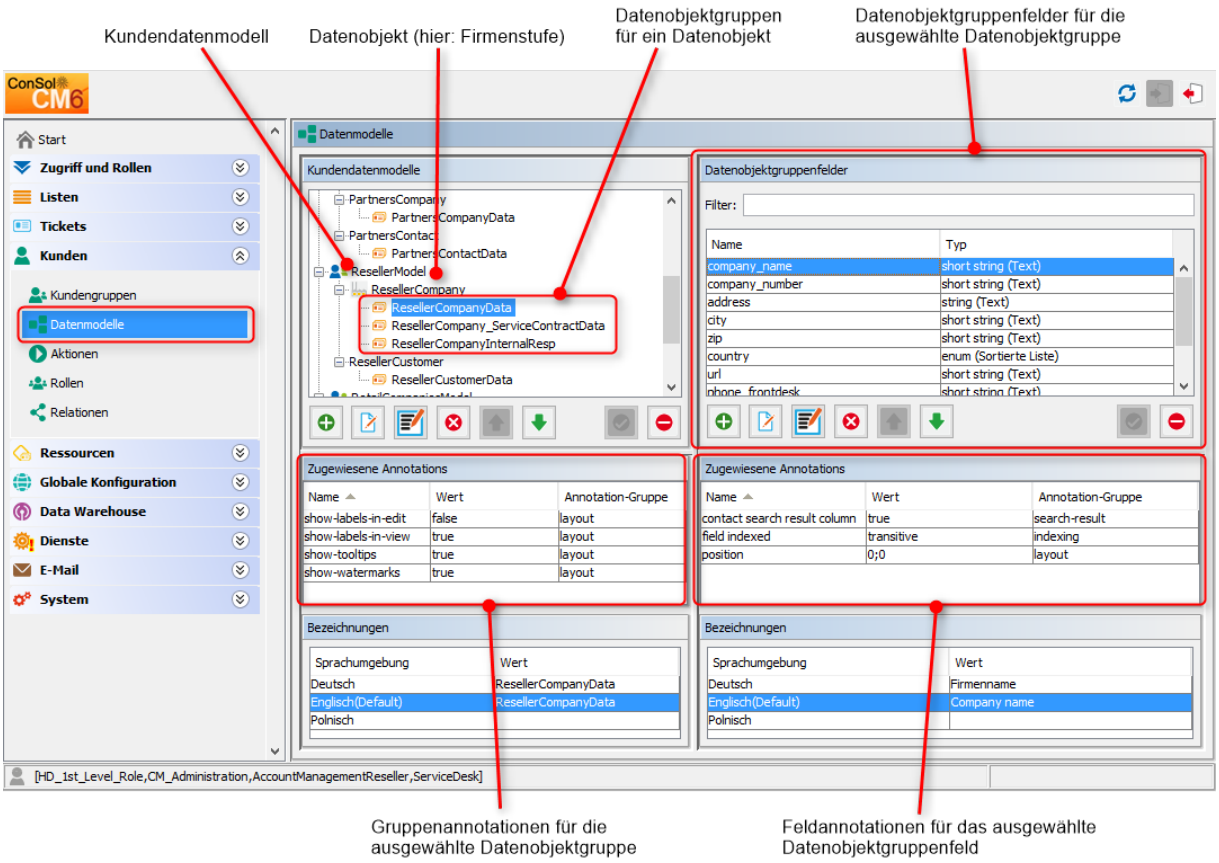


Abbildung 138: *ConSol CM Admin Tool - Verwaltung von Datenobjektgruppenfeldern für Kundendaten (CM-Version 6.10)*

Alles, was Sie in Version 6.9 über Datenobjektgruppenfelder gelernt haben, gilt auch für CM-Version 6.10. Einige Methoden wurden allerdings als veraltet (deprecated) gekennzeichnet.

In CM-Version 6.9 ist es noch möglich, Unit-Methods zu verwenden, die den Namen der Datenobjektgruppe nicht als Parameter haben, z. B.

```
Unit.getField(String pFieldName)
```

Dies funktioniert nur dann reibungslos, wenn ein Datenobjektgruppenfeld in allen Kundengruppen, auf die das Skript angewendet wird, den gleichen Namen hat, oder wenn im restlichen Code sichergestellt wird, dass es zur Laufzeit nicht in mehrdeutigen Situationen verwendet wird. In Version 6.10 sind die Unit-Methoden ohne den Namen der Datenobjektgruppe veraltet (deprecated) und werden in Version 6.11 und höher nicht mehr unterstützt. Die folgende Tabelle enthält alle diese Methoden:

CM-Versionen bis 6.9, deprecated in 6.10, nicht in 6.11	CM-Versionen 6.10 und höher
getField(String pFieldName)	getField(String pGroupName, String pFieldName)
getFieldValue(String pFieldName)	getFieldValue(String pGroupName, String pFieldName)
setFieldValue(String pFieldName, Object pFieldValue)	setFieldValue(String pGroupName, String pFieldName, Object pValue)
removeField(String pFieldName)	removeField(String pGroupName, String pFieldName)

D.8.6 Ressourcendaten

Ab Version 6.10.0 kann ConSol CM das optionale Modul *CM.Resource Pool* haben. Mit diesem Modul können Sie die CM-Datenbank erweitern und Ressourcenobjekte speichern. Dies können unterschiedliche Objekte wie IT-Assets, Verträge, Produkte oder andere in der entsprechenden Firma benötigte Objekte sein. Ähnlich wie das Kundendatenmodell wird auch das Ressourcendatenmodell mit dem Admin Tool definiert. Das Modell wird in der Navigationsgruppe *Ressourcen*, Navigationselement *Datenmodelle* definiert. Um in Workflow-Skripten mit Ressourcendaten arbeiten zu können, sollten Sie erst ein fundiertes Wissen des Funktionsprinzips des Ressourcenpools und der Einrichtung des Datenmodells erwerben. Lesen Sie dazu zuerst den Abschnitt über CM.Resource Pool im *ConSol CM Administratorhandbuch*.

Ähnlich wie bei Ticket- und Kundendaten werden auch die Ressourcendaten in Ressourcenfeldgruppen, die Ressourcenfelder enthalten, gespeichert.

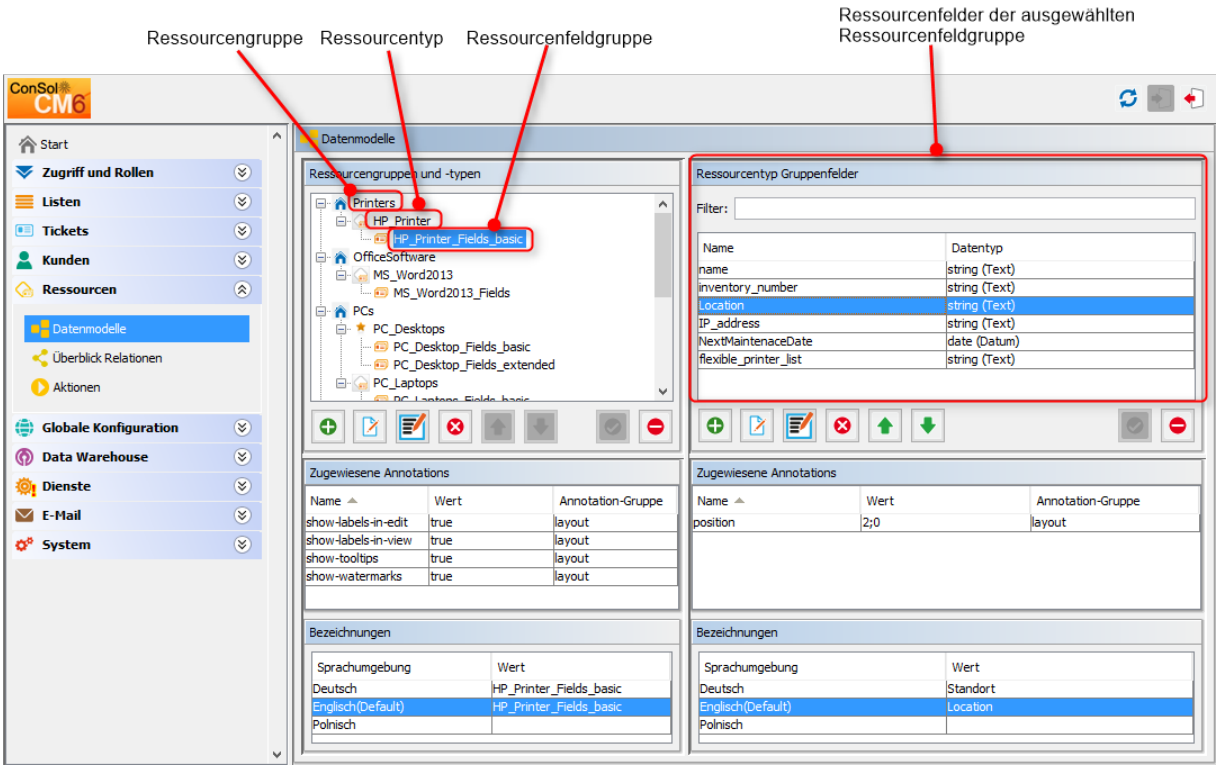


Abbildung 139: *Admin Tool: Definition des Ressourcendatenmodells*

Um Daten aus Ressourcenfeldern abzurufen, können Sie Methoden, wie die in den folgenden Beispielen gezeigten, verwenden.

```
// Display info about HP Printer, priting infos about resource fields into
 server.log

def my_resource_name = resource.get("HP_Printer_Fields_basic.name")
println 'Displaying resource information ...'
println ' Name is : ' + my_resource_name

def my_resource_inv_number = resource.get("HP_Printer_Fields_basic.inventory_
 number")
println ' Inventory number is : ' + my_resource_inv_number
```

Code-Beispiel 53: *Einfaches Ressourcenaktionsskript, das Informationen über die Ressource in der Log-Datei anzeigt*

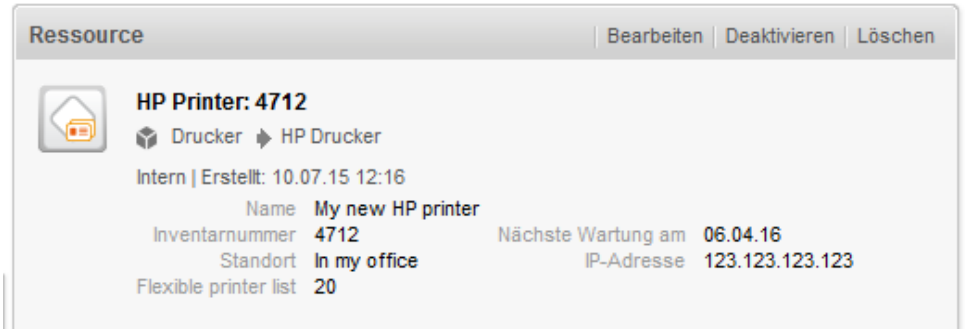


Abbildung 140: *Web Client: Ressourcenseite einer Ressource des Typs HP-Drucker*

```
stdout] [Susan-] Displaying resource information ...
stdout] [Susan-]  Name is : My new HP printer
stdout] [Susan-]  Inventory number is : 4712
```

Abbildung 141: *Log-Ausgabe des Skripts für den Drucker aus dem GUI-Beispiel*

In echten Prozessen sind die Anforderungen normalerweise komplexer. Es ist häufig erforderlich, alle Ressourcen eines bestimmten Typs abzurufen, die über eine bestimmte Art von Relation mit einem Ticket oder einer Ressource verknüpft sind. Dieses Thema wird detailliert im Abschnitt Arbeiten mit Ressourcenrelationen.

Weitere Programmierbeispiele aus dem ConSol CM Action Framework finden Sie im *ConSol CM Administratorhandbuch*, Abschnitt *Skripte für das Action Framework*.

D.8.6.1 Ein- und Ausblenden von Ressourcenfeldgruppen

Eine Ressourcenfeldgruppe kann mit einer workflowApi-Methode eingeblendet (sichtbar) und ausgeblendet (unsichtbar) werden. Dies funktioniert sowohl für Ressourcenfeldgruppen im Kopfbereich der Ressourcenseite als auch für Ressourcenfeldgruppen, die in den Tabs des Gruppenbereichs angezeigt werden.

Die Ressource *HP Printer* enthält im folgenden Beispiel eine Ressourcenfeldgruppe mit einer Ticketliste. Jedes Wartungsticket, das für die Ressource mit einer Ressourcenaktion erzeugt wird (eine detaillierte Erklärung zu Ressourcenaktionen finden Sie im *ConSol CM Administratorhandbuch*), soll zu dieser Liste hinzugefügt werden. Am Anfang ist die Ressourcenfeldgruppe als unsichtbar konfiguriert (Gruppenannotation *group-visibility = false*). Erst, wenn die Aktion durchgeführt wurde, soll die Ressourcenfeldgruppe auf der Ressourcenseite sichtbar werden.

Der folgende Code ist ein Auszug aus dem Ressourcenaktionsskript, derselbe oder ähnlicher Code kann aber auch in Workflow-Skripten verwendet werden.

```
// set ticket number in list in resource, newtic is the newly created maintenance
 ticket
def newtic_id = newtic.id.toString()
def newtic_name = newtic.name

//resource is the HP printer from which the action script is started
resource.add("HP_Printer_MaintenanceTickets.MaintenanceTicketsList",new Struct
 ().set("MaintenanceTicketID", newtic_id)
          .set("MaintenanceTicketName", newtic_name)
)
if (!executionContext) {
   return actionScriptResultFactory.getPostAction(PostActionType.FAILURE,
    "action.fail.wrong.activity")
}
// if ticktes are in the list in the resource, the field group should be visible
def groupName = "HP_Printer_MaintenanceTickets"
def fieldGroupDefinition = fieldDefinitionService.getGroupByName(groupName)
if (fieldGroupDefinition == null) {
   throw new IllegalArgumentException("There is no group definition with name '" +
    groupName + "'.")
}
resource.getGroupsConfiguration().setProperty(fieldGroupDefinition,
 GroupPropertyType.VISIBLE, "true")
```

Code-Beispiel 54: *Auszug aus einem Skript, das auf einer Ressourcenseite Zeilen zu einer Ticketliste hinzufügt und die Ressourcenfeldgruppe, die die Liste enthält, sichtbar macht*

D.8.7 Verwenden von Datenfeldern für (unsichtbare) Variablen

Manchmal ist es erforderlich, mit Variablen zu arbeiten, die nicht für Benutzerdefinierte Felder, Datenobjektgruppenfelder oder Ressourcenfelder verwendet werden und nicht auf der GUI sichtbar sind, die aber als Container für interne Programmiervariablen benötigt werden.

Diejenigen, die wissen, wie ConSol CM5-Workflows programmiert werden, kennen diese Container als *globale Variablen*. In ConSol CM6 können Sie das gleiche Ziel erreichen, indem Sie normale Benutzerdefinierte Felder (für Ticketdaten), Datenobjektgruppenfelder (für Kundendaten) oder Ressourcenfelder (für Ressourcendaten) mit dem erforderlichen Datentyp erstellen und diese auf *unsichtbar* setzen. Dies erfolgt über die Verwendung der Annotation *visibility = none*. Sie können die Variable auch während der Entwicklung des Prozesses sichtbar lassen, um den Wert des Feldes zu kontrollieren. Wenn das System an die QA und die Benutzer übergeben wird, können Sie sie dann auf unsichtbar setzen.

D.9 Senden von E-Mails

In diesem Kapitel werden folgende Themen behandelt:

D.9.1 Einführung in das Senden von E-Mails

Die Fähigkeit, E-Mails zu empfangen und zu senden ist eine Kernfunktion von ConSol CM. Eine detaillierte Einführung in die E-Mail-Funktionen finden Sie im *ConSol CM Administratorhandbuch*.

In diesem Abschnitt ist beschrieben, wie Sie Skripte schreiben, die E-Mails aus dem Workflow senden. Dies ist beispielsweise für folgende Anwendungsfälle sehr praktisch:

- Sie möchten eine automatische Empfangsbestätigung an den Kunden senden, der das Ticket erstellt hat.
- Sie möchten den Bearbeiter und seinen Vorgesetzten informieren, wenn die höchste Eskalationsstufe erreicht wurde.
- Sie möchten den Kunden darüber informieren, dass ein Problem gelöst wurde (und wie es gelöst wurde).

Normalerweise schreiben Sie den Text der E-Mail nicht in das Skript sondern arbeiten mit E-Mail-Vorlagen. Lesen Sie dazu zuerst den Abschnitt über den *ConSol CM Textvorlagen-Manager* im *ConSol CM Administratorhandbuch*.

D.9.2 Wichtige Methoden

D.9.2.1 ConSol CM-Version 6.9 und höher

Verwenden Sie ein Objekt der Klasse *Mail*.

Hier können Sie alle erforderlichen Parameter für eine E-Mail definieren und das *Mail-Objekt* so konfigurieren, dass es das queue-spezifische Standardskript für E-Mails verwendet. Dies ist das Skript, das die E-Mail verarbeitet, bevor sie das CM-System verlässt. Diese Art von Skript kann einer Queue zugewiesen werden (*E-Mail-Skript*, siehe Abschnitt *Queue-Verwaltung* im *ConSol CM-Administratorhandbuch*). Die Verwendung so eines Skripts kann hilfreich sein, wenn Sie zum Beispiel eine *Reply-To-Adresse* setzen möchten, die nicht die (in einer System-Property gespeicherte) Standardadresse für *Reply-To* ist.

D.9.3 Beispiele

D.9.3.1 Senden einer automatischen Empfangsbestätigung an den Kunden, der das Ticket erstellt hat.

ConSol CM Version 6.9 und höher

Dieses Skript kann an eine der ersten Aktivitäten des Workflows angehängt werden.

```
// create new mail object
def mail = new Mail()

// fetch main contact of the ticket
def maincontact = ticket.getMainContact()

// fetch e-mail address of the main contact. The Data Object Group Field has to be
 addressed using Data Object Group name:Data Object Group Field name
def toaddress = maincontact.get("MyCustomerDataObjectGroup:email")

// put the e-mail TO address into the Mail object
mail.setTo(toaddress)

// fetch the REPLY TO address, this is stored in a system property
def replyaddress = configurationService.getValue("cmweb-server-
 adapter","mail.reply.to")

// put the e-mail REPLY TO address into the Mail object
mail.setReplyTo(replyaddress)

// build e-mail text using a template which is stored in the Template Designer
def text = workflowApi.renderTemplate("Acknowledgement_of_receipt")

// put the e-mail text into the Mail object
mail.setText(text)

// create the subject of the e-mail, the ticket number with the correct Regular
 Expression has to be set for correct recognition of incoming e-mails for the
 ticket
def ticketname = ticket.getName()
def subject = "Your case has been registered as Ticket (" + ticketname + ")"

// put the subject into the Mail object
mail.setSubject(subject)

// send out the e-mail
mail.send()
```

Code-Beispiel 55: *Senden einer automatischen Empfangsbestätigung an den Kunden, der das Ticket erstellt hat, unter Verwendung eines Mail-Objekts*

D.9.3.2 Senden einer E-Mail an den Bearbeiter, wenn eine bestimmte Eskalationsstufe erreicht wurde

Dieses Skript kann an eine automatische Aktivität, die mit einem Zeit-Trigger verbunden ist, angehängt werden. Der Zeit-Trigger misst das Eskalationsintervall. Wenn die Deadline erreicht wurde, feuert der Trigger und das Ticket läuft in die automatische Aktivität.

ConSol CM Version 6.9 und höher

```
// create new mail object
def mail = new Mail()

// fetch current engineer of the ticket and set it as e-mail receiver
if (ticket.engineer){
   mail.setTargetEngineer(ticket.engineer)

   // fetch the REPLY TO address, this is stored in a system property
   def replyaddress = configurationService.getValue("cmweb-server-
    adapter","mail.reply.to")

   // put the e-mail REPLY TO address into the Mail object
   mail.setReplyTo(replyaddress)

   // build e-mail text using a template which is stored in the Template Designer
   def text = workflowApi.renderTemplate("ESCALATION_Mail")

   // put the e-mail text into the Mail object
   mail.setText(text)

   // create the subject of the e-mail, the ticket number with the correct Regular
    Expression has to be set for correct recognition of incoming e-mails for the
    ticket
   def ticketname = ticket.getName()
   def subject = "ESCALATION Level 3 REACHED! Ticket (" + ticket.getId() + ")"

   // put the subject into the Mail object
   mail.setSubject(subject)

   // send out the e-mail
   mail.send()
}
```

Code-Beispiel 56: *Senden einer E-Mail an den Bearbeiter, wenn eine bestimmte Eskalationsstufe erreicht wurde, unter Verwendung eines Mail-Objekts*

D.9.3.3 Senden einer E-Mail an einen Kunden unter Einbeziehung des queue-spezifischen E-Mail-Skripts

Dies ist das gleiche Skript, das auch im obigen Beispiel gezeigt ist. Es wird allerdings ein queue-spezifisches E-Mail-Skript angewendet. Eine detaillierte Erklärung über diese Art von Skript finden Sie im *ConSol CM Administratorhandbuch*, Abschnitt *Admin-Tool-Skripte*.

Dies bewirkt, dass die ausgehende E-Mail das Skript durchläuft, bevor sie das CM-System verlässt. E-Mail-Parameter wie *Cc*, *Bcc* oder *Reply-To* können geändert werden.

```
// create new mail object
def mail = new Mail()

// fetch main contact of the ticket
def maincontact = ticket.getMainContact()

// fetch e-mail address of the main contact. The Data Object Group Field has
// to be addressed using Data Object Group name:Data Object Group Field name
def toaddress = maincontact.get("MyCustomerDataObjectGroup:email")

// put the e-mail TO address into the Mail object
mail.setTo(toaddress)

// fetch the REPLY TO address, this is stored in a system property
def replyaddress = configurationService.getValue("cmweb-server-
 adapter","mail.reply.to")

// put the e-mail REPLY TO address into the Mail object
mail.setReplyTo(replyaddress)

// build e-mail text using a template which is stored in the Template Designer
def text = workflowApi.renderTemplate("Acknowledgement_of_receipt")

// put the e-mail text into the Mail object
mail.setText(text)

// create the subject of the e-mail, the ticket number with the correct Regular
 Expression
// has to be set for correct recognition of incoming e-mails for the ticket
def ticketname = ticket.getName()
def subject = "Your case has been registered as Ticket (" + ticketname + ")"

// put the subject into the Mail object
mail.setSubject(subject)

// Mail should use the e-mail script which is configured for the queue
mail.useDefaultScript()

// send out the e-mail
mail.send()
```

Code-Beispiel 57: *Senden einer E-Mail an einen Kunden unter Einbeziehung des queue-spezifischen E-Mail-Skripts, unter Verwendung eines Mail-Objekts*

D.9.3.4 Senden einer E-Mail an alle Kontakte des Tickets

Damit wird eine E-Mail mit allen Kunden (die eine E-Mail-Adresse haben) als Empfänger gesendet. Beachten Sie, dass dies ein einfaches Beispiel ist, das die Verwendung einer Liste zeigt. Die *Reply-To-Adresse* wird nicht gesetzt, sodass Antworten auf diese E-Mail nicht an das Ticket angehängt würden.

```
def custEmails = workflowApi.getContactList()*.get("email").findAll{it !=
 null}.join(",")
workflowApi.sendEmail(custEmails, "Confirmation", "Good afternoon, we received your
 request!", null, null)
```

Code-Beispiel 58: *Senden einer E-Mail an alle Kontakte des Tickets*

D.9.3.5 Senden einer E-Mail an jeden Kontakt in einer Liste aller Kontakte des Tickets

Damit wird eine E-Mail an jeden einzelnen Kunden (der eine E-Mail-Adresse hat) gesendet. Beachten Sie, dass dies ein einfaches Beispiel ist, das die Verwendung einer Liste zeigt. Die *Reply-To-Adresse* wird nicht gesetzt, sodass Antworten auf diese E-Mail nicht an das Ticket angehängt würden.

```
workflowApi.getContactList().each {
   def custEmail = it.get("email")
   if (custEmail){
      workflowApi.sendEmail(custEmail, "Confirmation",
      "Good afternoon, we received your request!", null, null)
   }
}
```

D.9.3.6 Senden einer E-Mail und Einfügen dieser in das Ticketprotokoll (CM 6.9 und höher)

Der folgende Code zeigt ein Skript, das aufgerufen wird, um eine Empfangsbestätigung an den Kunden zu senden, wenn ein Ticket für diesen Kunden erstellt wird. Wenn der Hauptkunde des Tickets eine Firma ist, wird keine E-Mail gesendet, weil die Firmen in unserem Beispielsystem keine E-Mail-Adressen haben. Im nächsten Schritt muss das Datenobjektgruppenfeld, das die E-Mail-Adresse enthält, gefunden werden. Der Feldname kann je nach Kundengruppe (mit dem entsprechenden Kundendatenmodell) unterschiedlich sein, was berücksichtigt werden muss. Wenn eine E-Mail gesendet wurde, wird dies in einem Eintrag im Ticketprotokoll mit dem richtigen Eintragstyp dokumentiert.

```
import com.consol.cmas.common.model.mail.Mail
import com.consol.cmas.common.model.content.MailEntryStatus;
import com.consol.cmas.common.model.content.AttachmentEntry;
import com.consol.cmas.common.model.content.ContentFile;
import com.consol.cmas.common.model.content.ContentEntryCategory
import com.consol.cmas.common.model.content.MailEntry;
import com.consol.cmas.common.util.MailHeadersUtil;
import com.consol.cmas.core.server.service.*;

// create new mail object
def mail = new Mail()
def ticket = workflowApi.getTicket()

// fetch main contatc of the ticket
def maincontact = ticket.getMainContact()
def unit_type = maincontact.definition.type.toString()

if(unit_type.equals('COMPANY')) {
   println 'No email address for company; no receipt notice sent.'
   return
} else if (unit_type.equals('CONTACT')){

   def toaddress_field
   def custgroup = maincontact.customerGroup.name
   println 'Customergroup is now ' + custgroup
   switch(custgroup) {
      case "Reseller": toaddress_field = "email";
      break;
      case "DirectCustomers": toaddress_field = "dir_cust_email"
      break;
      case "MyCustomerGroup": toaddress_field = "email"
      break;
      case "OurPartnerCompanies": toaddress_field = "email"
      break;
      case "RetailCustomers": toaddress_field = "retail_customer_email"
      break;
   }
   def toaddress = maincontact.get(toaddress_field)
   println 'toaddress is now ' + toaddress
   if (!toaddress){
      println 'No email address found for contact, no receipt notice sent.'
   } else {
      // put the e-mail TO address into the Mail object
      mail.setTo(toaddress)

      // fetch the REPLY TO address, theis is stored in a system property
      def replyaddress = configurationService.getValue("cmweb-server-
       adapter","mail.reply.to")
      // put the e-mail REPLY TO address into the Mail object
      mail.setReplyTo(replyaddress)

      // build e-mail text using a template which is stored in the Template Designer
      def text = workflowApi.renderTemplate("Acknowledgement_of_receipt")
```

```
      // put the e-mail text into the Mail object
      mail.setText(text)

      // create the subject of the e-mail, the ticket number with the correct
       Regular Expression
      // has to be set for correct recognition of incoming e-mails for the ticket
      def ticketname = ticket.getName()
      def subject = "Your case has been registered as Ticket (" + ticketname + ")"
      // put the subject into the Mail object
      mail.setSubject(subject)
      // Mail should use the e-mail script which is configured for the queue
      mail.useDefaultScript()
      // send out the e-mail and register status
      def attList = new ArrayList<AttachmentEntry>()
      def collection = new HashSet<MailEntry>()
      def mailStatus = true;

      // add attachments to attList .
      try {
         mail.send();
      } catch (Exception e){
         mailStatus = false;
      }

      MailEntry mailEntry = new MailEntry(subject,text);
      // you can also use an email template -> String text =
       workflowApi.renderTemplate
      mailEntry.setTicket(ticket);
      mailEntry.setCategory(ContentEntryCategory.OUTGOING_MAIL);
      // or other status -> see
       com.consol.cmas.common.model.content.ContentEntryCategory in API
      mailEntry.setCreationDate(new Date());
      mailEntry.setLastModificationDate(new Date());
      mailEntry.setMimeType("text/html"); // maybe "text/plain" shows linebreaks if
       "text/html" doesn't
      mailEntry.setEncoding("UTF-8");
      mailEntry.setAttribute(MailHeadersUtil.FROM_PROPERTY, replyaddress)
      mailEntry.setAttribute(MailHeadersUtil.TO_PROPERTY, toaddress)
      mailEntry.setAttribute(MailHeadersUtil.SUBJECT_PROPERTY,subject);

      if(mailStatus){
         mailEntry.setMailEntryStatus(MailEntryStatus.SUCCESS)
      } else {
         mailEntry.setMailEntryStatus(MailEntryStatus.FAILURE)
      }

      attList?.each { att ->
         mailEntry.addAttachment(att)
      }
      collection.add(mailEntry)
      workflowApi.updateTicket(ticket,collection)
   } // end if (!toaddress){
} // end of else if (unit_type.equals('COMPANY')){
```

Code-Beispiel 59: *Admin-Tool-Skript zum Senden einer Empfangsbestätigung und Einfügen der E-Mail als Eintrag in das Ticketprotokoll*

D.9.3.7 Arbeiten mit einer Vorlage für den Ticketnamen im Betreff der E-Mail

Im vorangegangenen Programmierbeispiel werden folgende Zeilen verwendet:

```
def ticketname = ticket.getName()
def subject = "Your case has been registered as Ticket (" + ticketname + ")"
```

Dies hat den Grund, dass das Muster für den E-Mail-Betreff sicherstellt, dass eine eingehende E-Mail dem richtigen Ticket zugewiesen werden kann. Dies bedeutet, dass Sie, wenn Sie die im obigen Beispiel gezeigte Codierung verwenden, sicherstellen müssen, dass das verwendete Muster mit dem Muster im Admin Tool (Navigationselement *E-Mail*) übereinstimmt (siehe auch folgende Abbildung).

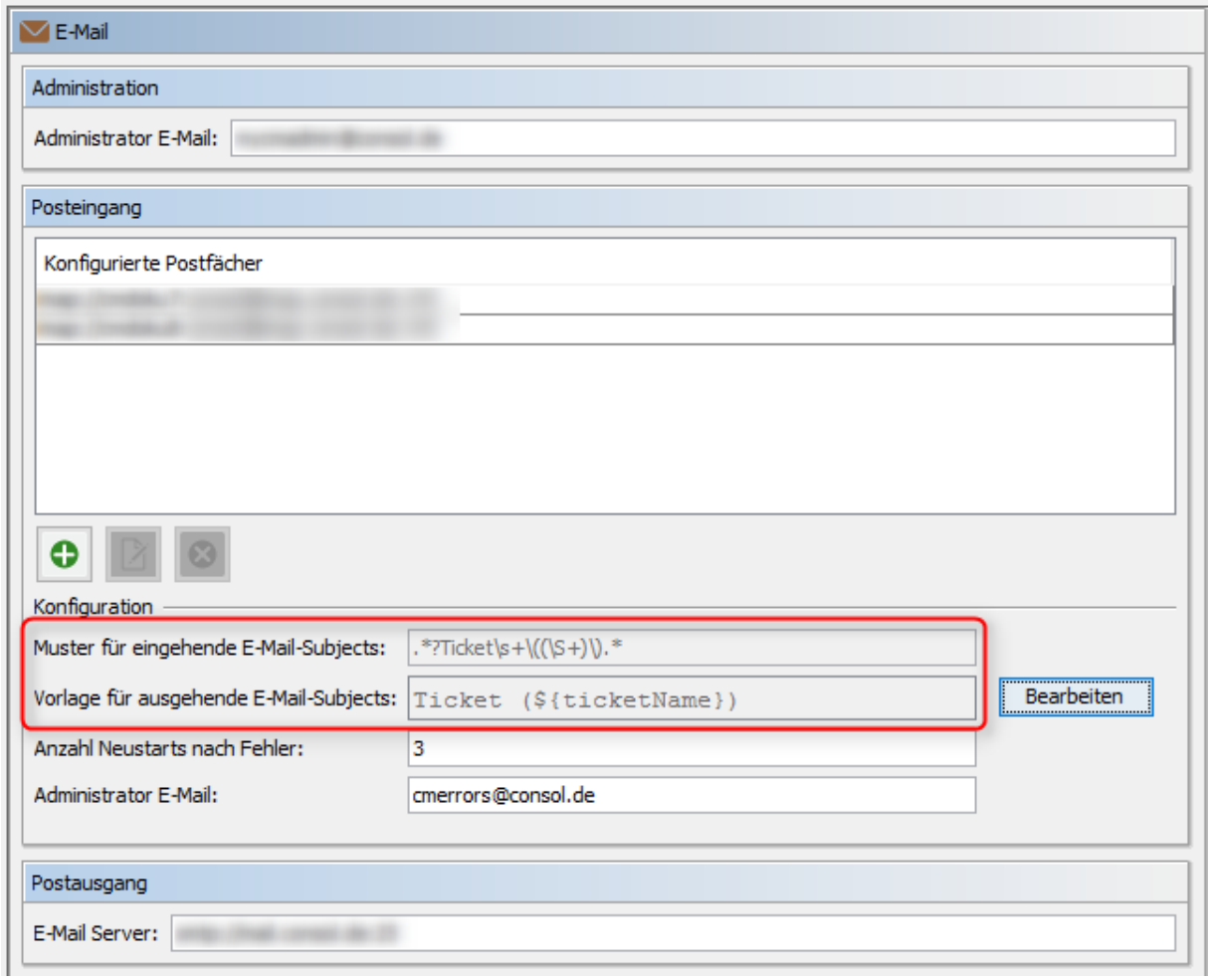


Abbildung 142: *Admin Tool: Muster und Vorlage für den E-Mail-Betreff*

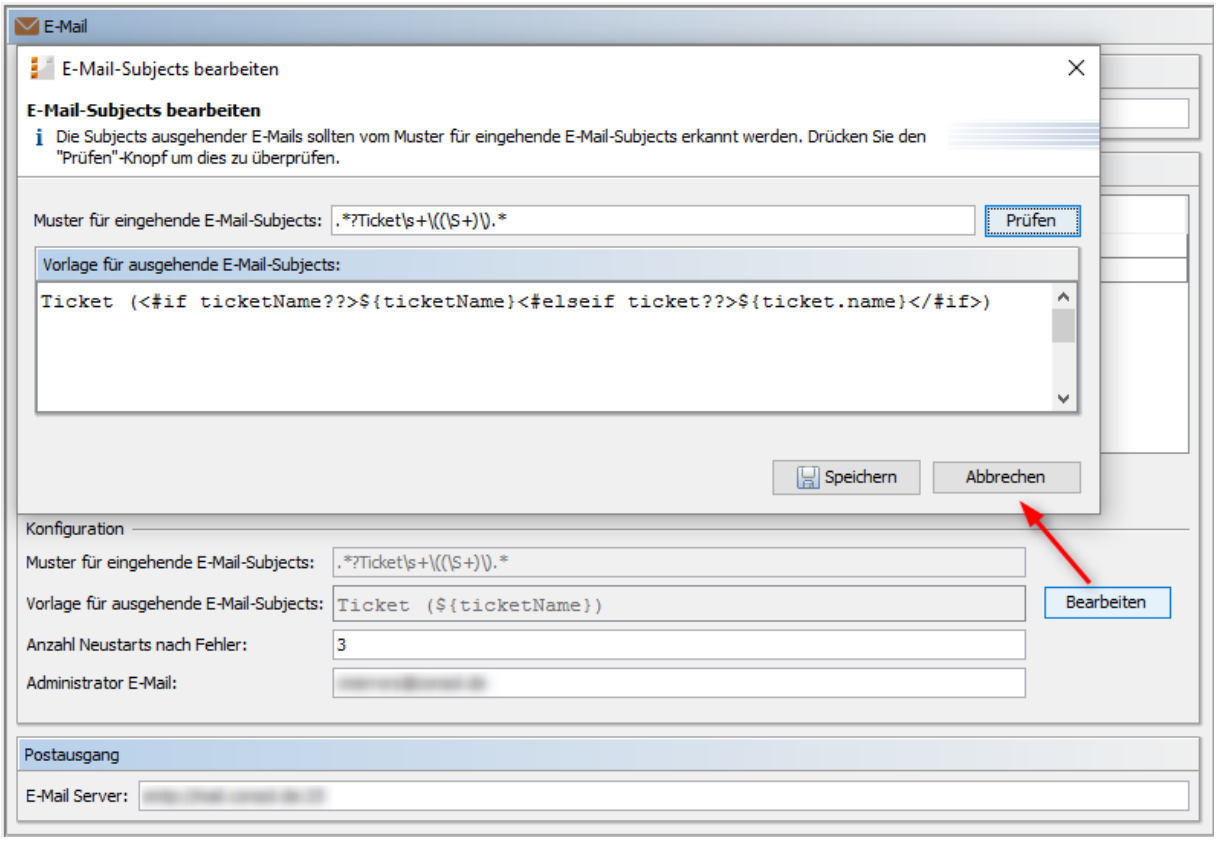
Um Inkonsistenzen zwischen dem Workflow-Code und der Systemkonfiguration (Admin Tool) zu vermeiden, können Sie auch mit dem implizit in CM vorhandenen Vorlagennamen arbeiten.

Verwenden Sie im Admin-Tool-Skript folgenden Code:

```
import static com.consol.cmas.common.util.TemplateUtil.TICKET_SUBJECT_TEMPLATE_NAME
( ... )
def subject = templateService.merge(TICKET_SUBJECT_TEMPLATE_NAME,
 [ticketName:ticket.name])
```

Beachten Sie, dass Sie einen *ticketName* genannten Parameter im Kontext setzen müssen, damit die Vorlage funktioniert.

Alternative Lösung: Wenn Sie die Vorlage für die ausgehende E-Mail so ändern, dass sie entweder *ticketName* oder das Objekt *ticket* annimmt (dann können Sie mit *ticket.name* arbeiten), können Sie auch mit der normalen Methode *renderTemplate()* arbeiten, die unten gezeigt ist.



Die Vorlage für ausgehende E-Mail-Betreffs ist dann:

```
Ticket (<#if ticketName??>${ticketName}<#elseif ticket??>${ticket.name}</#if>)
```

Und der Betreff lautet dann:

```
def subject = workflowApi.renderTemplate(TICKET_SUBJECT_TEMPLATE_NAME)
```

D.9.4 Schreiben von E-Mails aus Skripten, wenn Bearbeitervertretungsregeln gelten

Wenn Sie Skripte schreiben, in denen E-Mails an Bearbeiter und/oder Kunden gesendet werden, müssen Sie die Vertretungsregeln der Bearbeiter berücksichtigen. Detaillierte Erklärungen über die Vertretungsfunktion finden Sie im ConSol CM Administratorhandbuch (Abschnitt *Rollenverwaltung*) und im ConSol CM Benutzerhandbuch (Abschnitt *Bearbeiterprofil*).

Wichtige Informationen über die Konfigurationen von Vertretungen

Beachten Sie, dass es zwei unterschiedliche Szenarien für das Senden von E-Mails gibt und dass das Verhalten des CM-Systems bezüglich des Sendens von Vertretungs-E-Mails in den beiden Szenarien unterschiedlich sein kann!

1. Ein Bearbeiter schreibt eine E-Mail mit dem Ticket-E-Mail-Editor: Die Vertretungsregel wird angewendet und der Vertreter erhält eine Kopie der E-Mail. Dies bedeutet, dass alle E-Mails, die manuell mit CM gesendet werden, an den ursprünglichen Empfänger und an den aktuellen Vertreter gesendet werden. Das CM-System überprüft, ob für eine bestimmte E-Mail-Adresse eine Vertretungsregel aktiv ist! Bedenken Sie dies, wenn Sie die Vertretungsberechtigungen im Admin Tool konfigurieren und informieren Sie Ihre CM-Benutzer (Bearbeiter) über dieses Verhalten. Es kann unerwünschte Auswirkungen haben, insbesondere, wenn Personen als Bearbeiter und als Kontakte im ConSol CM-System erfasst sind (z. B. für einen internen Helpdesk).
2. Eine E-Mail wird automatisch aus dem CM-System gesendet: Es hängt von der spezifischen Konfiguration des CM-Systems ab, welche Bearbeiter eine Kopie der E-Mail erhalten. Die E-Mail wird **nicht** automatisch an die Vertreter gesendet! Es kann implementiert werden, dass der Vertreter eine Kopie erhält, dies ist aber nicht obligatorisch. Die ursprüngliche E-Mail kann aus einem Workflow-Skript oder einem Admin-Tool-Skript gesendet werden (das ebenfalls aus dem Workflow aufgerufen werden kann). Es hängt von der Implementierung dieses Skripts ab, wer eine Kopie der E-Mail erhält. Details dazu finden Sie im folgenden Abschnitt.

Für mittels Skripten gesendete E-Mails hängt das Verhalten des CM-Systems bezüglich der Vertretungsregeln von der Methode ab, die zum Senden der E-Mail verwendet wird.

Alle drei hier aufgeführten Methoden gehören zur Java-Klasse *Mail*.

D.9.4.1 setTo(String pTo)

```
Mail.setTo(<e-mail of originalReceivingEngineer>)
```

Diese Methode erhält ein String-Objekt als Parameter. Dabei handelt es sich um die E-Mail-Adresse des Empfängers. Mit dieser Methode wird **keine** Kopie der ursprünglichen E-Mail an die Vertreter gesendet.

D.9.4.2 setTargetEngineer(Engineer pTargetEngineer)

```
Mail.setTargetEngineer(<current engineer of the ticket>)
```

Diese Methode erhält ein Engineer-Objekt als Parameter und sendet die ursprüngliche E-Mail an die E-Mail-Adresse des Bearbeiters (sofern gesetzt) und an den vertretenden Bearbeiter, der als Vertreter für die ursprüngliche E-Mail-Adresse registriert ist. Seien Sie damit vorsichtig und lesen Sie die Infobox oben!

D.9.4.3 setTargetEngineers(List<Engineer> pTargetEngineers)

Diese Methode erhält eine Liste von Engineer-Objekten als Parameter und sendet die ursprüngliche E-Mail an die E-Mail-Adresse aller Bearbeiter in der Liste (sofern die E-Mail-Adresse gesetzt ist) und an die Vertreter aller in der Liste enthaltenen E-Mail-Adressen der Bearbeiter. Seien Sie damit vorsichtig und lesen Sie die Infobox oben!

Alle drei Methoden werden gleich verwendet, d. h. Sie müssen nur eine dieser Methoden verwenden, um die Adresse des Empfängers in einem Skript zu setzen, das eine E-Mail senden soll. Im folgenden Beispiel wird die Methode Mail.setTargetEngineer() verwendet.

import com.consol.cmas.common.model.mail.Mail

def ticket = workflowApi.ticket

```
import com.consol.cmas.common.model.mail.Mail

// Read engineer login from CF
// engineer login could also be retrieved, e.g., using ticket.engineer or
 engineerService.current, depending on what you need
def engineerName = ticket.get("main_data_fields.next_mail_to_engineer")
def engineer
if (engineerName) {
   engineer = engineerService.getByName(engineerName)
}
def engineerMail = engineer?.email
if (engineerMail) {
   def subject = "Your info mail about ticket " + ticket.getName() + " Subject:' "
    + ticket.getSubject() + "'"

   // Retrieve e-mail text from template
   def text = workflowApi.renderTemplate("engineer_info")

   def mail = new Mail()
   mail.setTargetEngineer(engineer)
   mail.setSubject(subject)
   mail.setText(text)
   // Use outgoing mail script, queue-specific
   mail.useDefaultScript()

   try {
      mail.send()
   } catch (Exception e){
      mailStatus = false
}
```

Code-Beispiel 60: *Workflow- oder Admin-Tool-Skript zum Senden einer E-Mail an einen Bearbeiter, unter Verwendung der Vertretungsfunktion durch Einsatz der Methode setTargetEngineer()*

D.10 Arbeiten mit Pfadinformationen

In diesem Kapitel werden folgende Themen behandelt:

D.10.1 Einleitung

Wie eine Datei im Dateisystems eines Computers kann auch jedes Element eines Workflows über den Pfad des Elements angesprochen werden. Dies ist möglicherweise erforderlich, wenn Sie innerhalb eines Workflow-Skripts mit dem Element arbeiten möchten. Ein Pfad bildet die hierarchische Struktur des Workflows ab.

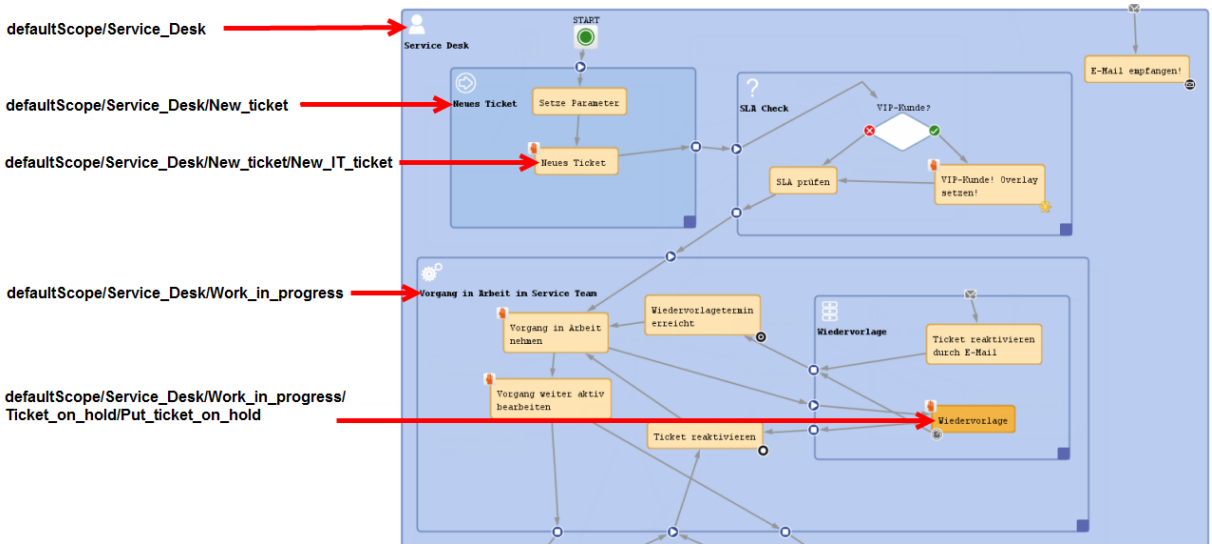


Abbildung 143: *ConSol CM Process Designer - Pfadinformationen (Beispiel: Aktivitäten und Bereiche)*

D.10.2 Abrufen von Pfadinformationen für ein Workflow-Element

Sie können den Pfad eines Elements kopieren, indem Sie mit der rechten Maustaste auf das Element klicken (in diesem Beispiel ein Adornment) und auf *Kopiere Pfad des Adornments in die Zwischenablage* klicken. Für Aktivitäten und Bereiche lautet der Menüeintrag *Kopiere Pfad des Knotens in die Zwischenablage*.

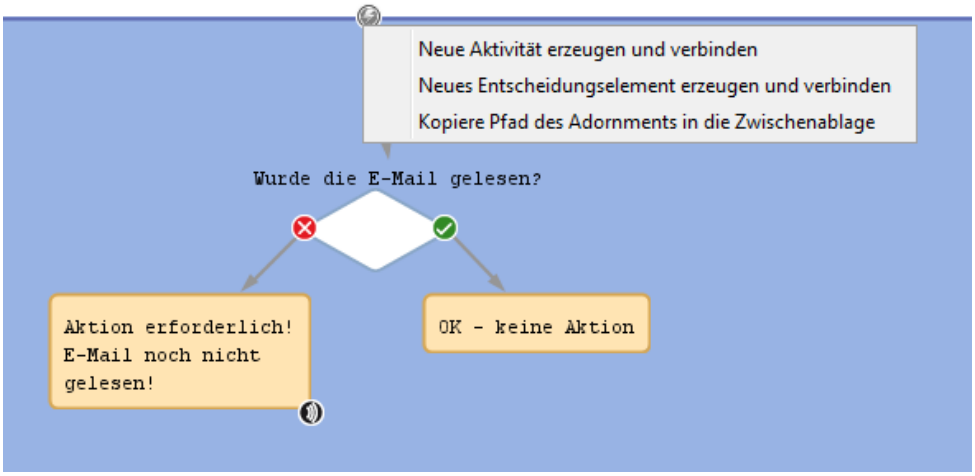


Abbildung 144: *ConSol CM Process Designer - Kopieren des Pfads eines Workflow-Elements*

D.10.3 Beispiele für die Verwendung von Pfadinformationen

D.10.3.1 Beispiel 1: Deaktivieren oder Reinitialisieren eines Zeit-Triggers

Ein typischer Fall für die Verwendung von Pfadinformationen ist die Reinitialisierung eines Zeit-Triggers, z. B. wenn Sie die Zeit nach Erhalt einer E-Mail messen möchten und sicherstellen möchten, dass sich ein Bearbeiter innerhalb von maximal 10 Minuten um die E-Mail kümmert. Dies bedeutet, dass Sie einen Zeit-Trigger immer wieder verwenden müssen und nach jeder E-Mail, die vom Ticket empfangen wurde, neu initialisieren müssen.

Wenn das Ticket erstellt wird, muss der Zeit-Trigger deaktiviert werden. Dazu wird folgender Code verwendet:

```
workflowApi.deactivateTimer("defaultScope/Service_Desk/TimeTrigger1")
```

Code-Beispiel 61: *Deaktivieren eines Zeit-Triggers*

Wenn eine E-Mail eingegangen ist, muss der Trigger reinitialisiert werden. Dazu wird folgender Code verwendet:

```
workflowApi.reinitializeTrigger("defaultScope/Service_Desk/TimeTrigger1")
```

Code-Beispiel 62: *Reinitialisieren eines Zeit-Triggers*

D.11 Arbeiten mit Kalendern und Zeiten

In diesem Kapitel werden folgende Themen behandelt:

D.11.1 Einleitung

Das Rechnen mit Daten und Zeiten spielt bei der Programmierung von ConSol CM-Workflows eine große Rolle. Für einen Zeit-Trigger (siehe Abschnitt Zeit-Trigger) muss der genaue Zeitpunkt, an dem er feuern soll, über das Skript gesetzt werden. Dadurch haben Sie verschiedene Möglichkeiten zur Steuerung von Eskalationszeiten, Erinnerungen für Bearbeiter und anderen *aktiven* Komponenten des ConSol CM-Prozesses. Beispiele für mögliche Berechnungen mit Daten und/oder Zeiten sind:

- Eskalationsdaten mit Zeit-Triggern
- *Datumsfelder* wie die gewünschte (oder erforderliche) Deadline

Wenn Sie ein Datum und/oder eine Zeit berechnen, müssen Sie entscheiden, ob ein Arbeitszeitkalender verwendet werden soll oder nicht. Ein Arbeitszeitkalender definiert die Arbeitszeiten für einen Prozess. Er wird mit dem Admin Tool definiert und einer oder mehrerer Queues zugewiesen.

Zum Beispiel kann das Servicedesk-Team Arbeitszeiten von 8:00 bis 18:00 an 6 Tagen der Woche haben, wohingegen das Verwaltungsteam an 5 Tagen pro Woche von 9:00 bis 17:00 arbeitet. Durch die Verwendung eines CM-Arbeitszeitkalenders wird sichergestellt, dass die Eskalation nicht in der Freizeit stattfindet und dass die Nicht-Arbeitsstunden nicht in die Berechnung der verstrichenen Eskalationszeit einfließen. Eine detaillierte Einführung in die Arbeitszeitkalender finden Sie im *ConSol CM Administratorhandbuch*.

Auf der anderer Seite gibt es Beispiele, bei denen ein Arbeitszeitkalender nicht erforderlich ist und stattdessen die *reine* Zeit auf Grundlage des normalen Kalenders verwendet werden soll. Zum Beispiel wenn Sie sich drei Wochen nach dem Erstkontakt bei einem Kunden melden sollen. Die folgenden Absätze zeigen Beispiele für beide Anwendungsfälle.

1 Tag bedeutet 24 Stunden absolute Zeit; dies hat nichts mit der Verwendung eines Kalenders zu tun. Der Kalender spielt nur eine Rolle, wenn der Zeit-Trigger aktiviert ist. Dann werden die 24 Stunden, d. h. 86400000 Millisekunden, als Eingabe für den Arbeitszeitkalender verwendet (sofern der Kalender aktiviert ist).

Beispiel:
Wenn der Zeitraum 1 Tag = 24 Stunden ohne Kalender ist, werden die 24 Stunden wie reguläre Zeit berechnet. Die Eskalation wird also einen Tag später zur gleichen Uhrzeit ausgelöst.
Im Gegensatz dazu: Wenn ein Kalender verwendet wird (mit zum Beispiel 7 Arbeitsstunden pro Arbeitstag), werden die 24 Stunden entsprechend dem Kalender aufgeteilt. Der Trigger feuert also über 3 Tage später (24 Stunden = 3 x 7 Stunden + 3 Stunden).

D.11.2 Rechnen mit Daten und Zeiten ohne CM-Arbeitszeitkalender

D.11.2.1 Beispiel: Setzen der Zeit eines Zeit-Triggers mit einem dynamischen Zeitraum

Der Zeit-Trigger für eine Eskalation wird abhängig von der Priorität konfiguriert:

```
// prio is 'medium'
def escalationTime = configurationService.getValue("custom-mycompany-
 properties","escalation.time.medium")
def escalationTimeMillisecs = escalationTime * 60 * 1000L
trigger.setDueTime( escalationTimeMillisecs )
```

Code-Beispiel 63: *Setzen der Zeit für einen Zeit-Trigger*

D.11.3 Rechnen mit Daten und Zeiten mit CM-Arbeitszeitkalender

D.11.3.1 Beispiel: Verwenden eines Zeit-Triggers mit einem Arbeitszeitkalender zum Berechnen der Eskalationszeit (CM 6.9)

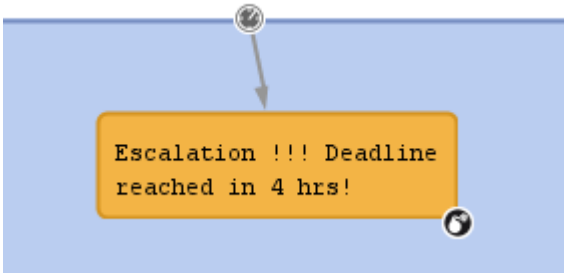


Abbildung 145: *ConSol CM Process Designer - Zeit-Trigger für eine Eskalation 4 Stunden vor der Deadline*

```
def deadl = ticket.get("serviceDesk_fields.desiredDeadline")

// 4hrs before deadline the escalation should be set
// business calendar should be used
// ServiceDeskCalendar is assigned to queue ServiceDesk, this is transparent here
def now = new Date()

// time required in millisecds
def four_hours = -4*60*60*1000L

// calculate escalation date
def escalDate = BusinessCalendarUtil.getBusinessTime(deadl, four_hours,
 ticket.queue.calendar)

// calculate and set due time

def dueTime = escalDate.time - now.time
trigger.setDueTime(dueTime)
```

Code-Beispiel 64: *Skript für einen Zeit-Trigger für eine Eskalation 4 Stunden vor der Deadline*

D.12 Arbeiten mit Objektrelationen

In ConSol CM können Sie mit drei Arten von Relationen arbeiten:

Relationstyp	Erklärung
Ticketrelationen	Hierarchische oder einstufige Relationen zwischen zwei Tickets, siehe Abschnitt Arbeiten mit Ticketrelationen.
Kundenrelationen	Relationen zwischen Kundendatenobjekten, d. h. Kontakten und Firmen, siehe Abschnitt Arbeiten mit Kundenrelationen (Datenobjektrelationen).
Ressourcenrelationen	Relationen zwischen einem Ressourcenobjekt und einem anderen Objekt. Das letztere kann vom Typ Ticket, Ressource oder Kunde sein. Siehe Abschnitt Arbeiten mit Ressourcenrelationen.

D.12.1 Arbeiten mit Ticketrelationen

In diesem Kapitel werden folgende Themen behandelt:

D.12.1.1 Einleitung

Relationen zwischen Tickets können Ihnen helfen, Ihren Geschäftsprozess sehr effizient zu modellieren.

ConSol CM bietet drei Arten von Relationen:

- **Einfache Ticketrelationen**
  Nicht hierarchisch, einfache Referenz. Jedes Ticket kann eine beliebige Anzahl an Referenzen haben.
  Eine einfache Ticketrelation kann von einem Bearbeiter im Web Client hergestellt werden oder von einem Programmierer über die ConSol CM-Programmierschnittstelle.
  In beiden Fällen kann eine Referenz nur zwischen zwei vorhandenen Tickets hergestellt werden.
- **Master-Slave-Relationen**
  Hierarchisch. Ein Master-Ticket kann mehrere Slave-Tickets haben. Ein Slave-Ticket hat immer genau ein Master-Ticket.
  Diese Struktur kann von einem Bearbeiter im Web Client hergestellt werden oder von einem Programmierer über die ConSol CM-Programmierschnittstelle.
  Eine Master-Slave-Relation kann nur zwischen zwei vorhandenen Tickets hergestellt werden, d. h. beide Tickets müssen zuerst existieren und dann kann eine Master-Slave-Relation hergestellt werden, um sie zu verbinden.
- **Parent-Child-Relationen**
  Hierarchisch. Ein Parent-Ticket kann mehrere Child-Tickets haben. Ein Child-Ticket hat immer genau ein Parent-Ticket.
  Diese Struktur kann nur über die ConSol CM-Programmierschnittstelle hergestellt und verändert werden.
  Eine Parent-Child-Relation kann zwischen vorhandenen Tickets hergestellt werden. Zudem kann ein neues Child-Ticket während des Prozesses erstellt werden.

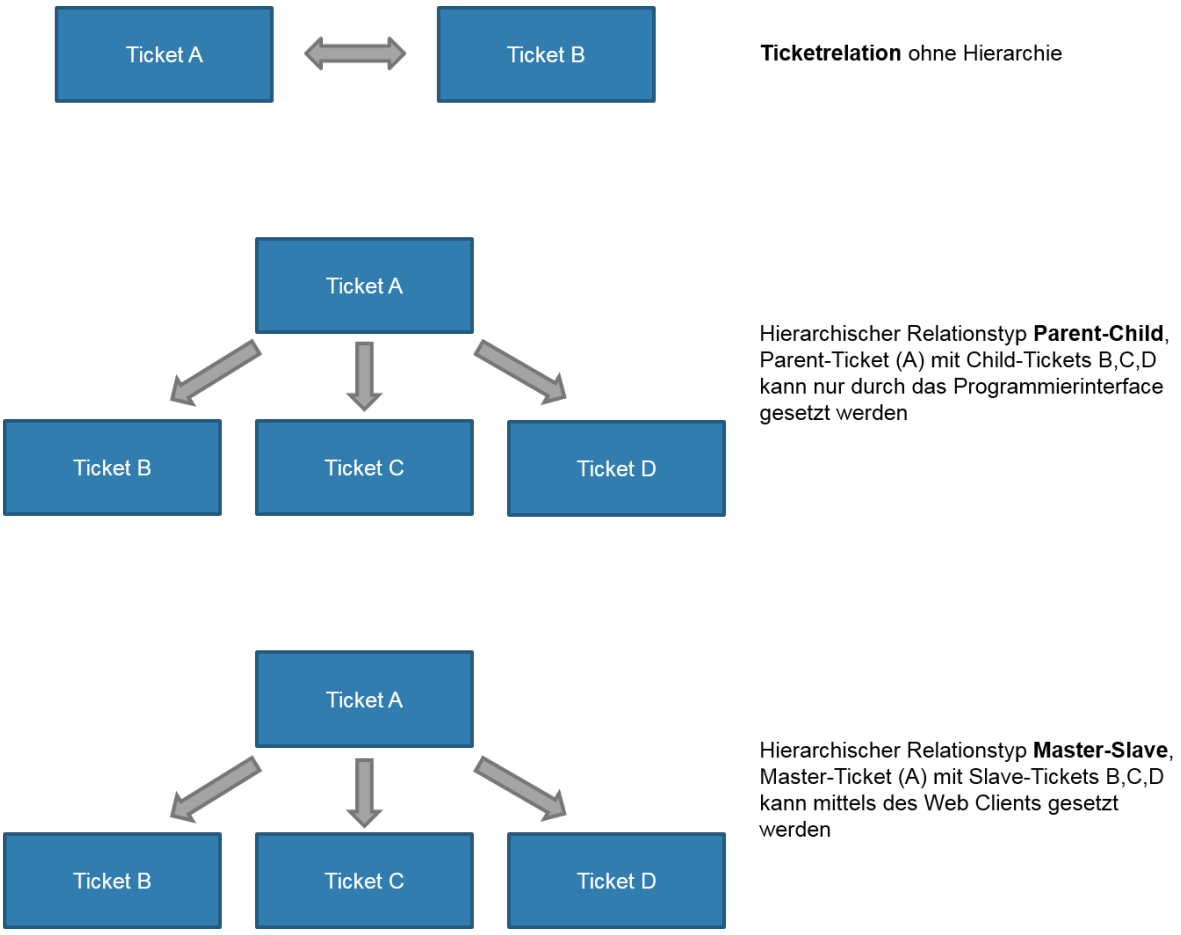


Abbildung 146: *ConSol CM-Relationstypen*

In diesem Abschnitt wird nicht erklärt, wie man Ticketrelationen im Web Client erstellt. Dies ist detailliert im *ConSol CM Benutzerhandbuch* beschrieben. Im vorliegenden Abschnitt zeigen wir Ihnen, wie man Relationen über die Programmierschnittstelle, in Workflow-Skripten, herstellt.

In der ConSol CM-Workflow-API wird der Referenztyp durch die Klasse (enum) `com.consol.cmas.common.model.ticket.TicketRelationType` abgebildet. Diese hat drei Werte:

- REFERENCE
- MASTER_SLAVE
- PARENT_CHILD

D.12.1.2 Einfache Ticketrelation ohne Hierarchie

Dieser Relationstyp kann hilfreich sein, wenn Sie Referenzen erzeugen möchten, die Ihnen helfen, die mit einem Ticket verbundenen Tickets einfacher zu finden als in einer Suche.

Beispiele für Anwendungsfälle sind:

- Wenn ein neues Ticket erstellt wird, möchten Sie sehen, ob es andere offene Tickets des gleichen Kunden gibt. Wenn ja, erstellen Sie eine Relation zwischen den Tickets. Auf diese Weise

kann ein Bearbeiter schnell von einem offenen Tickets des Kunden zum nächsten wechseln.

- Wenn ein neues Ticket für eine bestimmte Hardware-Kategorie erstellt wird, möchten Sie Referenzen zu allen anderen Tickets des gleichen Hardware-Typs herstellen.

Dieser Relationstyp kann entweder mit dem Web Client oder der Programmierschnittstelle hergestellt und geändert werden. Eine Relation des Typs *REFERENCE* kann also innerhalb eines Workflow-Skripts hergestellt werden und dann von einem Bearbeiter im Web Client geändert werden, sofern dieser die erforderlichen Zugangsberechtigungen hat.

Beispiel: Erstellen einer einfachen Relation zwischen zwei Tickets

```
workflowApi.addRelation(TicketRelationType.REFERENCE, "This is a very important
 relation", pSourceTicketId, pTargetTicketId)
```

Code-Beispiel 65: *Erstellen einer Ticketrelation des Typs REFERENCE mit der workflowAPI*

Syntax: Suchen aller referenzierten Tickets

ConSol CM-Versionen 6.10.4.0 und höher

```
List<Ticket> mytickets = workflowApi.getReferencedTickets()
```

D.12.1.3 Master-Slave-Relationen

Dieser Relationstyp kann hilfreich sein, wenn Sie eine Hierarchie zwischen einer bestimmten Anzahl an vorhandenen Tickets erzeugen möchten. Denken Sie daran, dass dieser Relationstyp sowohl im Web Client als auch über die Programmierschnittstelle hergestellt werden kann. An dieser Stelle wird allerdings nur der Programmieransatz erklärt.

Beispiele für Anwendungsfälle sind:

- In einer Firma gibt es mehrere Projekte, die jeweils durch ein Ticket abgebildet werden. Wenn die Entscheidung getroffen wird, eines der Projekte in ein übergreifendes Programm zu integrieren (das ebenfalls durch ein Ticket abgebildet wird), klickt der Projektmanager auf die Workflow-Aktivität *In Programm integrieren*. Dabei muss das richtige Programm ausgewählt werden (z. B. in einem ACF). Im Skript der Workflow-Aktivität *In Programm integrieren* wird das Programmticket als *Master-Ticket* des aktuellen Projekttickets gesetzt.
- In einem Service-Team werden verschiedene Produkte verwaltet. Für jedes Produkt gibt es ein Produktticket. Wenn ein neues Service-Ticket erstellt wird, klickt der Bearbeiter auf die Aktivität *Produkt setzen*, mit der er das verknüpfte Produkt aus einem Drop-down-Menü auswählen kann. Im Workflow-Skript der Aktivität *Produkt setzen* wird das Service-Ticket automatisch als *Slave* des Produkttickets gesetzt.

Eine Master-Slave-Relation kann entweder mit dem Web Client oder der Programmierschnittstelle hergestellt und geändert werden. Eine Relation des Typs *MASTER_SLAVE* kann also innerhalb eines Workflow-Skripts hergestellt werden und dann von einem Bearbeiter im Web Client geändert werden, sofern dieser die erforderlichen Zugangsberechtigungen hat. Verwenden Sie die Struktur *Parent-Child*, wenn Sie sicherstellen möchten, dass kein Bearbeiter die Tickethierarchie verändern kann.

Beispiel: Erstellen einer Master-Slave-Relation zwischen zwei Tickets

```
//in this script the project ticket (= current ticket) is set as slave ticket to
// the program ticket which becomes the master

// fetch the program ticket ID. The ID of the program ticket is already stored
// in a CF in the project (=current) ticket
def progTicketId = ticket.get("ReferencesFields.ProgramTicketId")

// fetch ID of current ticket (which will become the slave)
def mySlaveProjectId = ticket.id

workflowApi.addRelation(TicketRelationType.MASTER_SLAVE, "Slave Ticket: This
 project is part of the program indicated in the master ticket", progTicketId,
 mySlaveProjectId)
```

Code-Beispiel 66: *Erstellen einer Ticketrelation des Typs MASTER_SLAVE unter Verwendung von workflowAPI*

Syntax: Finden aller Slave-Tickets

ConSol CM-Versionen 6.10.3.x und niedriger

```
// the ticket can be set, might be current ticket or another ticket
List<Ticket> mytickets = workflowApi.getTargetTickets(myTicket.getId(),
 TicketRelationType.MASTER_SLAVE)
```

Code-Beispiel 67: *Version A: Finden aller Ziel-Tickets (hier: alle Slave-Tickets)*

```
// used for current ticket
List<Ticket> mytickets = workflowApi.getTargetTickets(TicketRelationType.MASTER_
 SLAVE)
```

Code-Beispiel 68: *Version B: Finden aller Ziel-Tickets (hier: alle Slave-Tickets)*

ConSol CM-Versionen 6.10.4.0 und höher

```
List<Ticket> mytickets = workflowApi.getSlaveTickets()
```

Code-Beispiel 69: *Finden aller Slave-Tickets des aktuellen Tickets*

Syntax: Finden des Master-Tickets

ConSol CM-Versionen 6.10.4.0 und höher

```
def mticket = workflowApi.getMasterTicket()
```

Code-Beispiel 70: *Finden des Master-Tickets des aktuellen Tickets*

D.12.1.4 Parent-Child-Relationen

Dieser Relationstyp kann hilfreich sein, wenn Sie eine Hierarchie zwischen einer bestimmten Anzahl an Tickets erzeugen möchten, die nicht manuell verändert werden kann.

Beispiele für Anwendungsfälle sind:

- Ein Projekt soll von einem Projektmanagement-Ticket verwaltet werden, das das Parent-Ticket ist. Alle Aufgaben innerhalb des Projekts werden als Child-Tickets abgebildet. Diese Struktur wird beim Einrichten des Projekttickets automatisch von einem Workflow-Skript erstellt.
- Es ist eine Systemmigration geplant, die ein Parent-Ticket verwendet. Für jede Komponente, die migriert werden muss, wird ein Child-Ticket erstellt. Diese Struktur wird beim Einrichten des Projekttickets automatisch von einem Workflow-Skript erstellt.

Der Relationstyp *PARENT_CHILD* kann nur über die Programmierschnittstelle hergestellt und geändert werden. Daher kann eine Relation dieses Typs in einem Workflow-Skript hergestellt werden und danach nur durch andere Skripte geändert werden.

Beispiel 1: Erstellen eines neuen Child-Tickets als Child des aktuellen Tickets

```
// this script creates a ticket for a task which will be child ticket
// of a project ticket (which will be the parent)

// create a new ticket, which will become the task (=child) ticket
Ticket newTask = new Ticket()

// fetch the subject of the parent-to-be ticket, i.e. of the current ticket
def subj = ticket.subject
// or longer: def subj = ticket.getSubject()

// set the subject of the new task (= child) ticket
newTask.setSubject("New Task for project " + subj)

// put the task (= child) ticket into the tasks queue
def tasksQueue = queueService.getByName("Tasks")
newTask.setQueue(tasksQueue)

// Initially, the new task ticket will not have an engineer
newTask.setEngineer(null)

// define the ticket text, i.e. the first comment in the new task ticket
def taskTicketText = "Please work on this task asap"

// the contact for the new task ticket should be the same as the one for the
 project ticket:
def taskContact = workflowApi.getPrimaryContact()

//create PARENT_CHILD relation between project (parent) and task (child)
workflowApi.createChildTicket(newTask, taskTicketText, taskContact)
```

Code-Beispiel 71: *Erstellen eines Child-Tickets*

Beispiel 2: Finden des Parent-Tickets eines Tickets

```
def my_parent = workflowAPI.getParentTicket()
```

Code-Beispiel 72: *Finden des Parent-Tickets eines Tickets*

Beispiel 3: Finden aller Child-Tickets eines Tickets

```
// only works for current ticket:
List<Ticket> my_childtickets = workflowApi.getChildTickets()
```

Code-Beispiel 73: *Finden aller Child-Tickets eines Tickets*

Beispiel 4: Finden aller Brother-Tickets (andere Child-Tickets) des gleichen Parent-Tickets

```
// only works for current ticket:
List<Ticket> my_brothers = workflowApi.getBrotherTickets()
```

Code-Beispiel 74: *Finden aller Brother-Tickets eines Child-Tickets*

D.12.1.5 Wichtige Methoden für die Arbeit mit Ticketrelationen

Beachten Sie folgende Regeln bei der Arbeit mit Ticketrelationen:

- Bei MASTER_SLAVE-Relationen ist der Master immer die Quelle.
- Bei PARENT_CHILD-Relationen ist der Parent immer die Quelle.
- Bei einfachen REFERENCE-Relationen ist die Quelle das Ticket, aus dem die Relation erstellt wird.

WorkflowApi-Methoden

Die folgenden Methoden sind Methoden der Klasse **WorkflowContextService**, die implizit als Objekt **workflowApi** in Workflow-Skripten verfügbar ist.

Methode von workflowApi (workflowContextService)	Erklärung
Ticket createChildTicket(Ticket pTicket, String pTicketText, Unit pCustomer)	Erstellt ein neues Child-Ticket. Queue, Priorität und Kategorie müssen richtig gesetzt werden.
List<Ticket> getChildTickets()	IntSet, das die Ticketobjekte der Child-Tickets des aktuellen Tickets enthält.
List<Ticket> getBrotherTickets()	IntSet, das die Ticketobjekte der Brother-Tickets des aktuellen Tickets enthält.
Ticket getParentTicket()	Ticket-Objekt des Parent-Tickets oder *null*, wenn das aktuelle Ticket kein Parent-Ticket hat.
List<Ticket> getTargetTickets (TicketRelationType pType)	Ruft eine Liste mit Ticket-Objekten ab, zu denen das aktuelle Ticket Relationen eines bestimmten Typs hat. Für diese Relationen ist das aktuelle Ticket das Quellticket.
List<Ticket> getTargetTickets (long pTicketId, TicketRelationType pType)	Ruft eine Liste mit Ticket-Objekten ab, zu denen das aktuelle Ticket Relationen eines bestimmten Typs hat. Für diese Relationen ist das mit *pTicketId* angegebene Ticket das Quellticket.
List<Ticket> getSourceTickets (TicketRelationType pType)	Ruft eine Liste mit Ticket-Objekten ab, zu denen das aktuelle Ticket Relationen eines bestimmten Typs hat. Für diese Relationen ist das aktuelle Ticket das Zielticket.

Methode von workflowApi (workflowContextService)	Erklärung
List<Ticket> getSourceTickets (long pTicketId, TicketRelationType pType)	Ruft eine Liste mit Ticket-Objekten ab, zu denen das aktuelle Ticket Relationen eines bestimmten Typs hat. Für diese Relationen ist das angegebene Ticket das Zielticket.
boolean hasTargetTickets (TicketRelationType pType)	Prüft, ob das Ticket Zieltickets hat. Prüft, ob es Relationen gibt, bei denen dieses Ticket das Quellticket ist.
boolean hasTargetTickets(long pTicketId, TicketRelationType pType)	Prüft, ob das angegebene Ticket Zieltickets hat. Prüft, ob es Relationen gibt, bei denen dieses Ticket das Quellticket ist.
boolean hasSourceTickets (TicketRelationType pType)	Prüft, ob das Ticket Quelltickets hat. Prüft, ob es Relationen gibt, bei denen dieses Ticket das Zielticket ist.
boolean hasSourceTickets(long pTicketId, TicketRelationType pType)	Prüft, ob das angegebene Ticket Quelltickets hat. Prüft, ob es Relationen gibt, bei denen dieses Ticket das Zielticket ist.
void changeSourceTickets (TicketRelationType pType, long pTargetTicketId, List<Long> pSourceTicketIds)	Für das Zielticket (z. B. ein Child-Ticket) werden die Relationen eines angegebenen Typs (z. B. PARENT_CHILD) entfernt. Für den gleichen Relationstyp wird eine neue Relation mit den angegebenen Quelltickets erstellt.
void changeTargetTickets (TicketRelationType pType, long pSourceTicketId, List<Long> pTargetTicketsIds)	Für das angegebene Quellticket werden alle Relationen des angegebenen Typs entfernt. Für die Liste der angegebenen Zieltickets werden neue Relationen des angegebenen Typs erstellt.
void removeRelation(TicketRelationType pType, long pSourceTicketId, long pTargetTicketId)	Entfernt die Ticketrelation zwischen zwei Tickets mit dem angegebenen Typ.
void addRelation(TicketRelationType pType, String pComment, long pSourceTicketId, long pTargetTicketId)	Fügt eine Relation des angegebenen Typs zwischen Ticket *sourceTicketId* und *targetTicketId* hinzu.
Ticket getMasterTicket()	Verfügbar ab CM-Version 6.10.4.0 Gibt das Master-Ticket des aktuellen Tickets zurück.
List<Ticket> getSlaveTickets()	Verfügbar ab CM-Version 6.10.4.0 Gibt eine Liste aller Slave-Tickets des aktuellen Tickets zurück.

Methode von workflowApi (workflowContextService)	Erklärung
List<Ticket> getReferencedTickets()	Verfügbar ab CM-Version 6.10.4.0 Gibt eine Liste aller Tickets zurück, die eine einfache Referenzrelation zum aktuellen Ticket haben.

Tabelle 1: *Wichtige Methoden von workflowAPI für die Arbeit mit Ticketrelationen*

TicketRelationService Methods

Wenn Sie im Admin Tool mit Skripten arbeiten (die dann aus einem Workflow-Skript aufgerufen werden), ist workflowApi nicht verfügbar. Sie können Methoden der Klasse *TicketRelationService*, die als Singleton *ticketRelationService* verfügbar ist, verwenden.

Methode von ticketRelationService	Erklärung
List<TicketRelation> getByTicket(Ticket pTicket, TicketRelationDirection pTicketRelationEnd, TicketRelationType... pRelationType)	Ruft eine Liste aller Ticketrelationen für ein angegebenes Ticket ab, eingeschränkt durch den Relationstyp und optional das Ende, an dem das Ticket angehängt werden soll.
Set<TicketRelation> getByTickets (Set<Ticket> pTickets, TicketRelationDirection pTicketRelationEnd, TicketRelationType... pRelationType)	Ruft den Satz aller Ticketrelationen für einen angegebenen Satz an Tickets ab, eingeschränkt durch den Relationstyp und optional das Ende, an dem die Tickets angehängt werden sollen.

Tabelle 2: *Wichtige Methoden von TicketRelationService für die Arbeit mit Ticketrelationen*

Beispiel mit Methoden aus workflowApi und ticketRelationService

```
// use wflApi:
println 'Displaying slave tickets from wfl script ...'
List<Ticket> slave_tics = workflowApi.getSlaveTickets()
slave_tics?.each(){ st ->
   println ' Slave Ticket is now ' + st.getId() + ' -- ' + st.getSubject()
}
```

Code-Beispiel 75: *Beispielskript, zeigt IDs und Namen der Slave-Tickets an, Workflow-Version*

```
// DisplaySlaveTickets.groovy
// use in AT:
import com.consol.cmas.common.service.*
import com.consol.cmas.common.model.ticket.TicketRelation
import com.consol.cmas.common.model.ticket.TicketRelationDirection
import com.consol.cmas.common.model.ticket.TicketRelationType
println 'Displaying slave tickets from AT script ...'
def ticket = workflowApi.getTicket()

List<TicketRelation> t_rel = ticketRelationService.getByTicket(ticket,
 TicketRelationDirection.ANY, TicketRelationType.MASTER_SLAVE)
t_rel?.each(){ tr ->
   println 'Source ticket is now ' + tr.sourceTicket.id + ' -- ' +
    tr.sourceTicket.subject
   println 'Target ticket is now ' + tr.targetTicket.id + ' -- ' +
    tr.targetTicket.subject
}
```

Code-Beispiel 76: *Beispielskript, zeigt IDs und Namen der Slave-Tickets an, Admin-Tool-Skriptversion*

```
scriptExecutionService.execute("DisplaySlaveTickets.groovy")
```

Code-Beispiel 77: *Aufrufen des obigen Admin-Tool-Skripts aus einer Workflow-Aktivität*

D.12.2 Arbeiten mit Kundenrelationen (Datenobjektrelationen)

In diesem Kapitel werden folgende Themen behandelt:

D.12.2.1 Einleitung

Seit Version 6.9.0 bietet ConSol CM *Kundenrelationen*. In älteren Versionen ist diese Funktion nicht verfügbar.

Um mit Kundenrelationen zu arbeiten, müssen Sie ein fundiertes Wissen über *FlexCDM*, das flexible Kundendatenmodell von ConSol CM, haben. Eine detaillierte Einführung dazu finden Sie im *ConSol CM Administratorhandbuch (Version 6.9)*.

Drei Objekte sind entscheidend:

Objekt	Java-Klasse	Admin-Tool-Beschreibung	Erklärung
Kunde	Unit	<keine>	Die allgemeine Beschreibung oder das allgemeine Objekt, das einen Kunden darstellt, d. h. eine Person oder Firma, die in der CM-Datenbank erfasst ist.
Firma	Unit	Datenobjekt des Typs *Firma*	Ein Objekt auf der Firmenstufe (d. h. die oberste Stufe im Kundenmodell). Das kann eine echte Firma, eine Maschine oder ein anderes Objekt, das diese Stufe darstellt, sein. Ein Objekt auf der *Firmenstufe* kann die *übergeordnete Stufe* für ein Objekt auf der *Kontaktstufe* sein. Aus logischer Sicht kann eine Firma mehrere Kontakte haben.
Kontakt	Unit	Datenobjekt des Typs *Kontakt*	Ein Objekt auf der Kontaktstufe (d. h. die unterste Stufe im Kundenmodell). Das kann eine echte Person oder ein anderes Objekt, das diese Stufe darstellt, sein. Ein Objekt auf der *Kontaktstufe* kann ein eigenständiges Objekt (in einem einstufigen Kundenmodell) sein, oder zu einem Objekt auf der *Firmenstufe* gehören. Aus logischer Sicht kann ein Kontakt zu keiner oder genau einer Firma gehören.

⚠ Bedenken Sie, dass der Hauptkunde eines Tickets ab CM-Version 6.9 ein Kontakt oder eine Firma sein kann! Die verwendete Methode ist *ticket.getMainContact()*. Sie gibt ein Objekt der Klasse *Unit* zurück. Das Objekt kann entweder ein Kontakt oder eine Firma sein!

Kundenrelationen stellen Relationen zwischen Kunden, d. h. Firmen und Kontakten dar.

Diese können sein:

- **gerichtet**
  unterschiedliche Ebenen in einer Hierarchie
- **Referenz**
  gleiche Ebene, keine Hierarchie

Eine Relation hat einen der folgenden Typen:

- **Firma - Firma**
  z. B. ... *hat eine Kooperation mit* ... (Firma X arbeitet mit Firma Y zusammen)
    - Die Firmen können zur gleichen oder zu unterschiedlichen Kundengruppen gehören.
    - Die beteiligten Kundengruppen können das gleiche oder unterschiedliche Kundendatenmodelle haben.
- **Firma - Kontakt**
  z. B. ... *ist Kunde von* ... (Kontakt X ist Kunde von Firma Y)
    - Die Firma und der Kontakt können zur gleichen oder zu unterschiedlichen Kundengruppen gehören.
    - Die beteiligten Kundengruppen können das gleiche oder unterschiedliche Kundendatenmodelle haben.
- **Kontakt - Kontakt**
  z. B. ... *wird betreut von* ... (Kontakt X von Firma X wird von Kontakt Y von Firma Y betreut)
    - Die Firmen und Kontakte können zur gleichen oder zu unterschiedlichen Kundengruppen gehören.
    - Die beteiligten Kundengruppen können das gleiche oder unterschiedliche Kundendatenmodelle haben.

In der Programmierschnittstelle wird ein Kundenobjekt (d. h. ein Kontakt oder eine Firma) durch ein Objekt der Klasse *Unit* abgebildet.

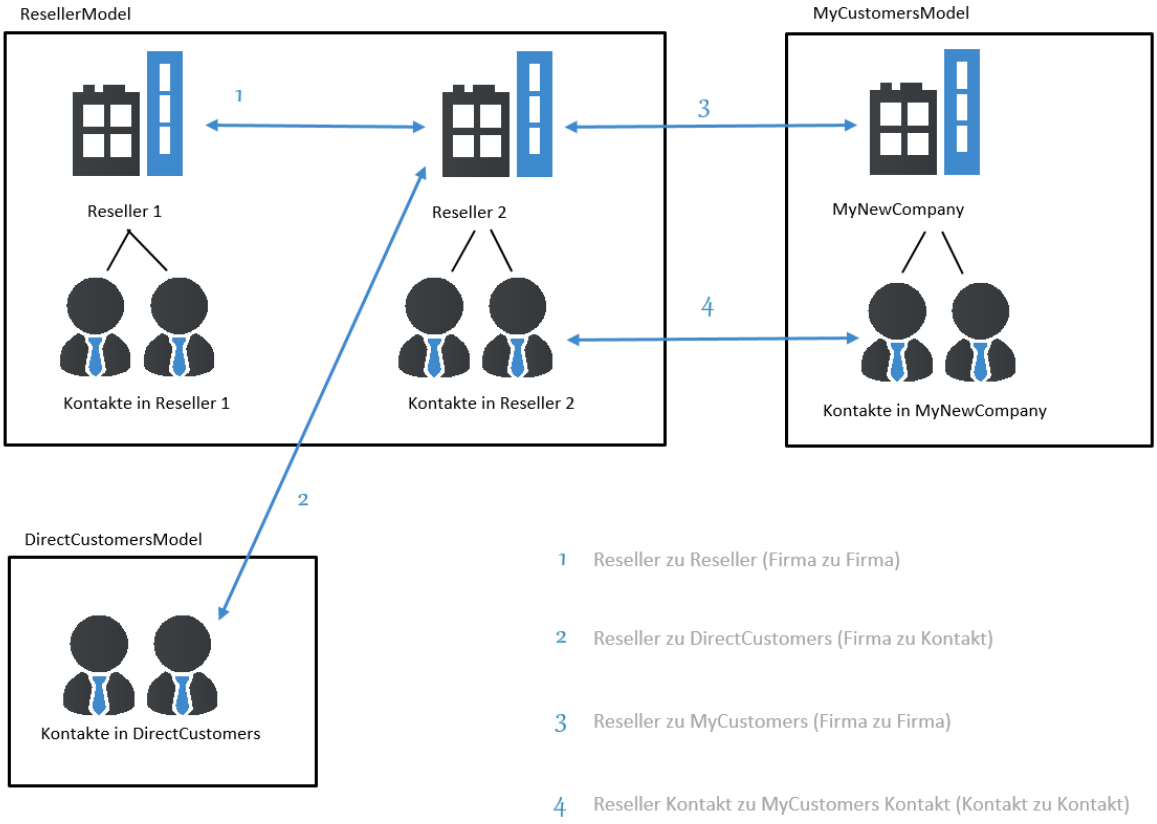


Abbildung 147: *Kundenrelationen in ConSol CM*

⚠ Um in Workflow-Skripten mit *Unit-Relationen* arbeiten zu können, müssen Sie alle erforderlichen Relationen zuerst im Admin Tool erstellt und konfiguriert haben, bevor Sie mit der Programmierung beginnen.

D.12.2.2 Erstellen von Unit-Relationen über die Programmierschnittstelle

⚠ In diesem Handbuch verwenden wir manchmal die neuen Begriffe *Datenobjekt* und *Datenobjektdefinition*, die zum neuen Kundenmodell von ConSol CM-Version 6.9 und höher (*FlexCDM*) gehören. Die Namen der entsprechenden Java-Klassen sind allerdings immer noch *Unit* und *UnitDefinition*. Alle anderen Java-Klassen, die Kundendatenobjekte behandeln, heißen ebenfalls noch *Unit...* . Bedenken Sie dies, wenn Sie als Administrator oder Programmierer mit einer Version ab 6.9.x arbeiten. Details dazu finden Sie in der *ConSol CM-Java-API-Dokumentation*.

Beispiel: Reseller-Endkunde-Relation hinzufügen

Im folgenden Beispiel wurde im Admin Tool eine Relation definiert, die eine Relation zwischen *Reseller und Endkunde* darstellt. Eine Firma der Kundengruppe *Reseller* verkauft Produkte an einen Kunden (eine Person, ein Kontakt) der Kundengruppe *DirectCustomers*.

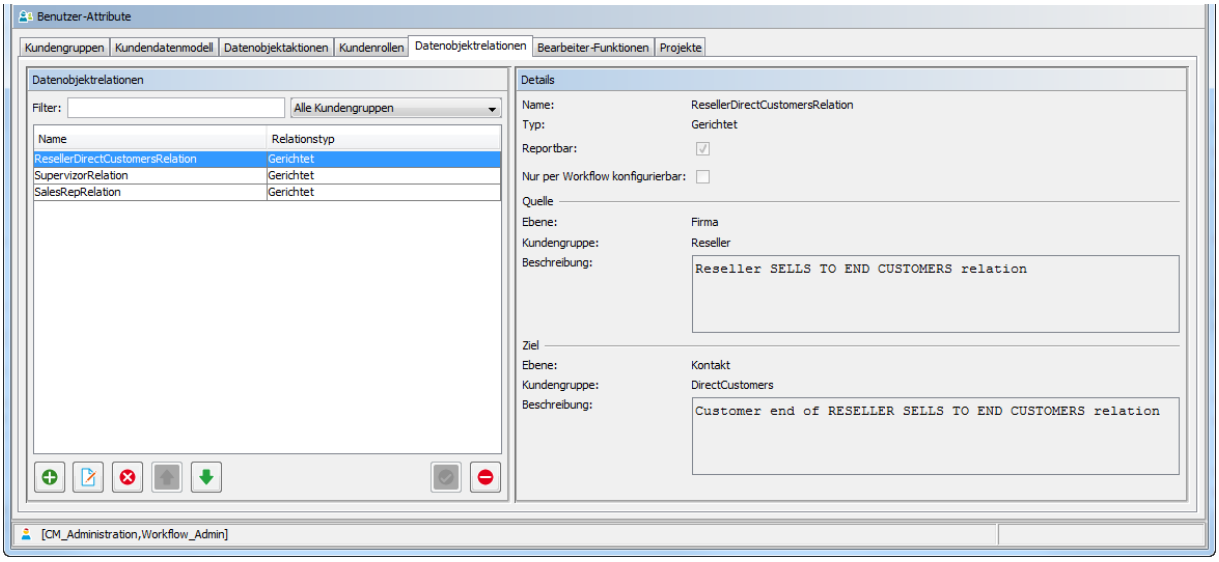


Abbildung 148: *ConSol CM Admin Tool - Definition der Reseller-Endkunde-Relation*

Es wird ein Ticket mit einem Hauptkunden erstellt. Der Kunde ist ein Mitarbeiter der Reseller-Firma. Der Endkunde, an den die Reseller-Firma Produkte verkauft, wird als Zusatzkunde mit der Rolle *Endkunde* zum Ticket hinzugefügt. Der Bearbeiter, der an dem Ticket arbeitet, soll in der Lage sein, über eine Workflow-Aktivität eine Relation zwischen der Reseller-Firma (Quelle) und dem Endkunden (Ziel) zu erstellen.

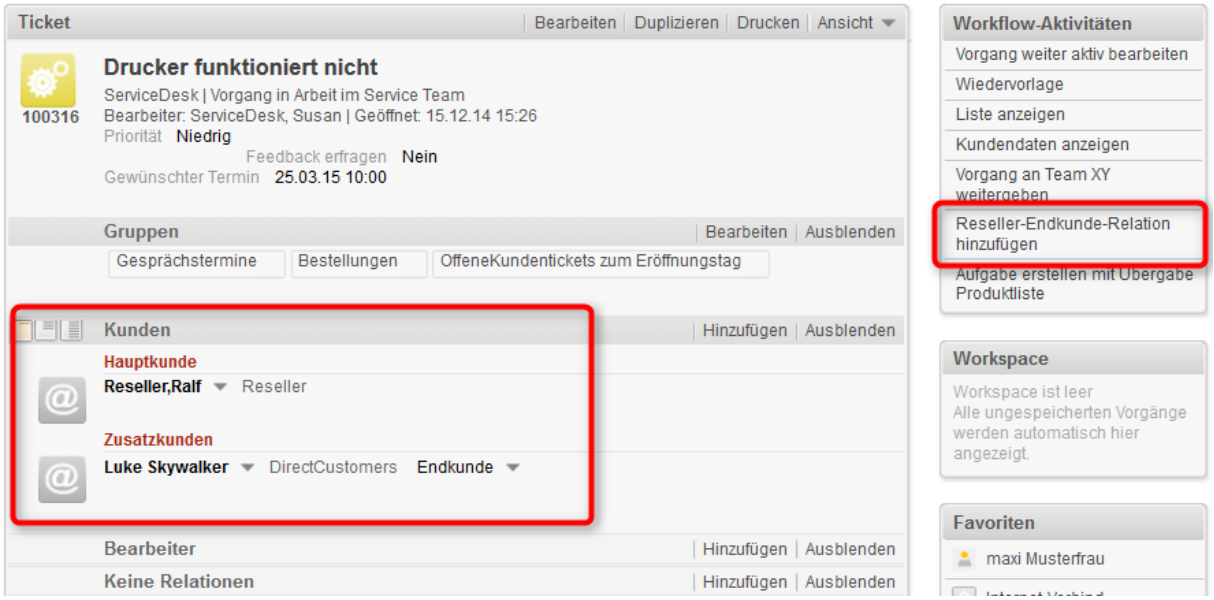


Abbildung 149: *ConSol CM Web Client - Beispielticket mit dem Hauptkunden und einem Zusatzkunden*

Im Workflow *Service Desk* gibt es eine Workflow-Aktivität *Reseller-Endkunde-Relation hinzufügen* (siehe folgende Abbildung).

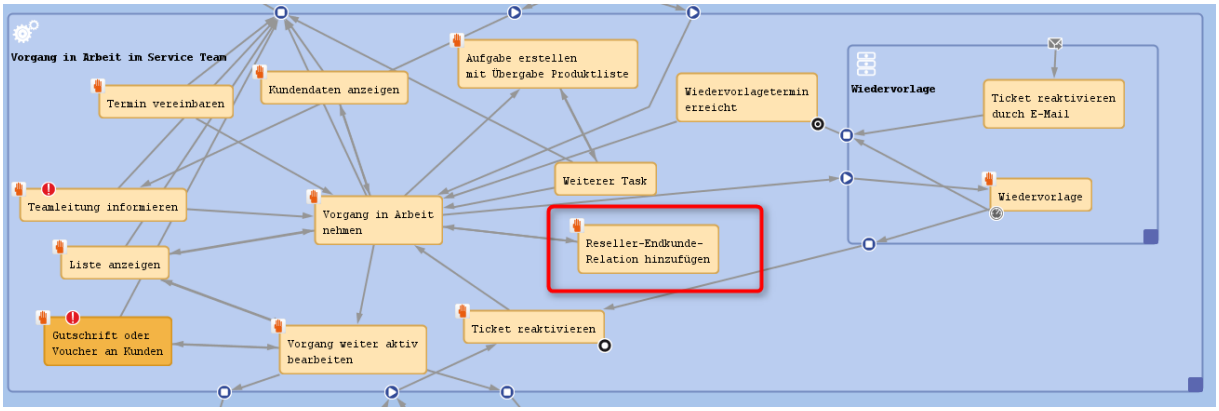


Abbildung 150: *ConSol CM Process Designer - Workflow-Aktivität zum Hinzufügen einer Unit-Relation*

Das folgende Skript wird in der Workflow-Aktivität *Reseller-Endkunde-Relation hinzufügen* verwendet:

```
// get Company of the main customer of the ticket, this is the RESELLER company:
// 1. get the main contact of the ticket. Here, this is a person = contact:
def cont = ticket.getMainContact()
// 2. get the company of the contact, this is the reseller company
def comp = cont.getCompany()

// get all additional contacts of the ticket in the customer role „end customer"
//and start the loop for all those additional customers:
def end_custs = ticket.getContacts("end customer").each() { e_cust ->

   //build all components for new unit relation:
   // 1.get the UnitDefinition by name (this is the name used in the Admin Tool):
   def unitrel_def = unitRelationDefinitionService.getByName
    ("ResellerDirectCustomersRelation")
   // create a new unit relation object with the unit definition and source
   // (the reseller company) and target (the end customer person)
   def new_rel = new UnitRelation(unitrel_def, comp, e_cust, "This Reseller sells
    to the end customer")

   // create the new unit relation in the system
   def new_rel2 = unitRelationService.create(new_rel)
}
```

Code-Beispiel 78: *Hinzufügen einer Datenobjektrelation mithilfe eines Workflow-Skripts*

Wenn der Bearbeiter die Workflow-Aktivität ausgeführt hat, wird eine Relation von der *Reseller-Firma* zum *Endkunden* hinzugefügt.

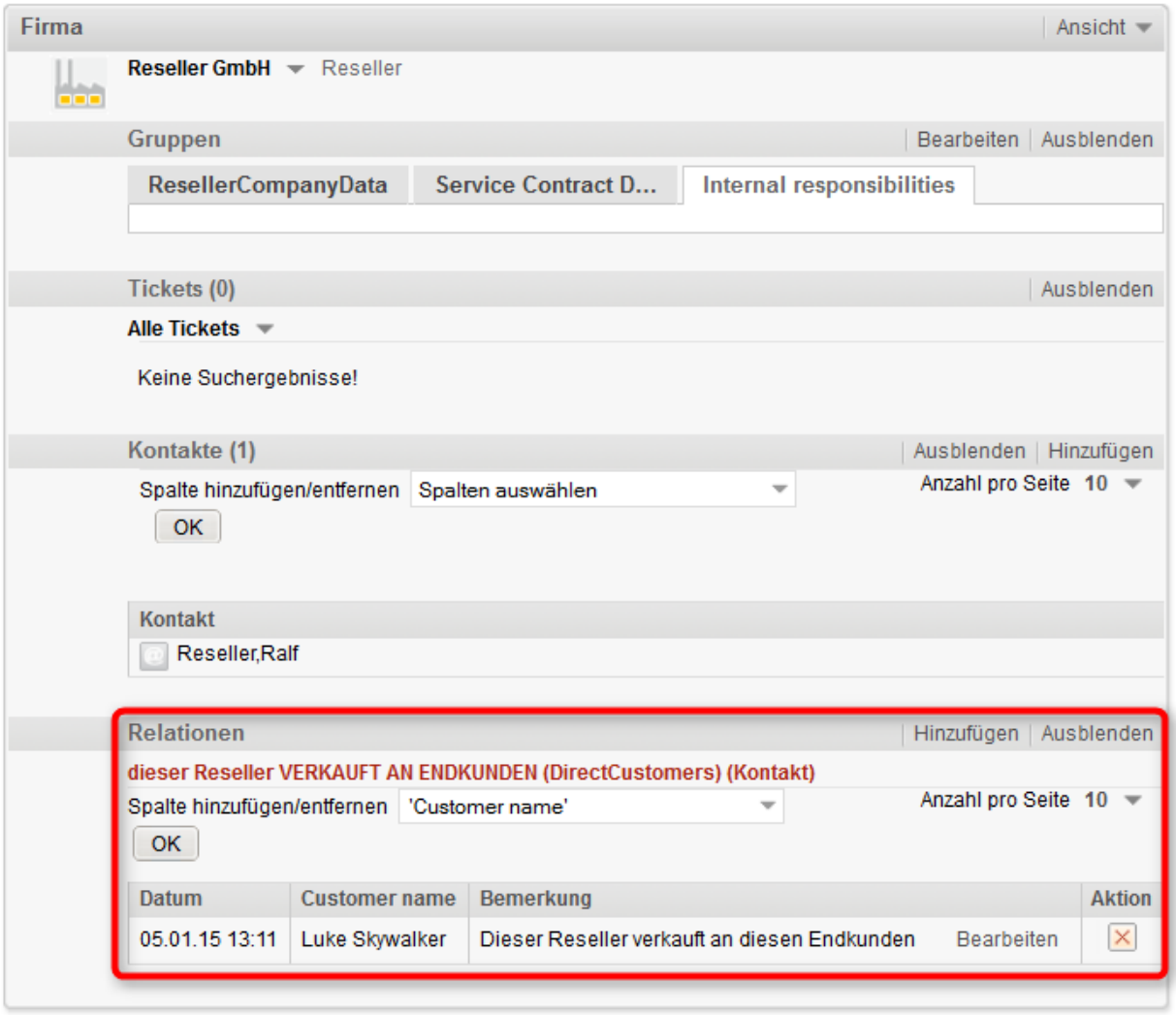


Abbildung 151: *ConSol CM Web Client - Neue Unit-Relation (erzeugt durch Workflow-Skript)*

D.12.2.3 Wichtige Java-Klassen für die Arbeit mit Unit-Relationen

Java-Klasse	Erklärung
`Unit`	Ein Datenobjekt (Unit): ein Kontakt oder eine Firma.
`UnitRelation`	Eine Relation zwischen zwei Datenobjekten (Units). Sichtbar im Web Client auf der Kontakt- oder Firmenseite unter *Verknüpfte Firmen und Kontakte*.

Java-Klasse	Erklärung
`UnitRelationDefinition`	Die Definition einer Unit-Relation wie sie im Admin Tool unter *Benutzer-Attribute - Datenobjektrelationen* (CM-Version 6.9) oder *Kunden - Relationen* (CM-Version 6.10) konfiguriert ist. Eine `UnitRelation` hat immer eine bestimmte `UnitRelationDefinition`.
`UnitRelationDefinitionService`	Singleton. Verfügbar als Objekt `unitRelationDefinitionService`. Service, der hilfreiche Methoden für die Arbeit mit Datenobjektrelationen (Unit-Relationen) bietet. Details dazu finden Sie in der *ConSol CM-Java-API-Dokumentation*.
`UnitRelationService`	Singleton. Verfügbar als Objekt `unitRelationService`. Service, der hilfreiche Methoden für die Arbeit mit Datenobjektrelationen (Unit-Relationen) bietet. Details dazu finden Sie in der *ConSol CM-Java-API-Dokumentation*.

D.12.3 Arbeiten mit Ressourcenrelationen

In diesem Kapitel werden folgende Themen behandelt:

D.12.3.1 Einleitung

Seit Version 6.10.0 bietet ConSol CM das Modul CM.Resource Pool und damit die Möglichkeit, Relationen zwischen Ressourcenobjekten und anderen Objekten herzustellen. Eine Ressourcenrelation kann eine Ressource mit folgenden Objekten verbinden:

- einem Ticket
- einem Kunden (Kontakt oder Firma, d. h. einer Unit)
- einer anderen Ressource

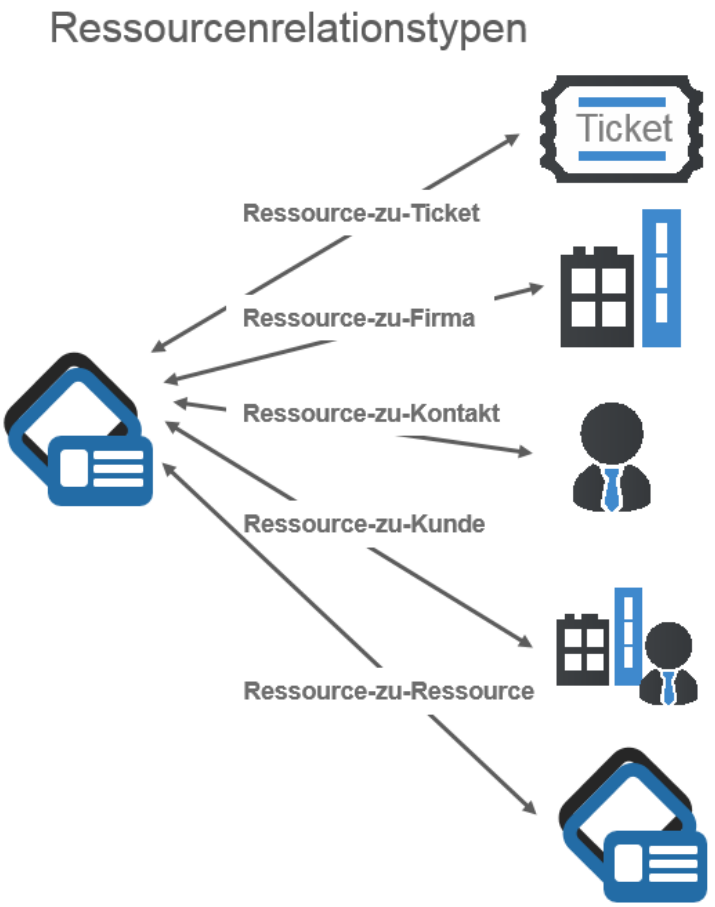


Abbildung 152: *Arten von Ressourcenrelationen*

Um mit Ressourcenrelationen arbeiten zu können, müssen Sie ein fundiertes Wissen des Ressourcendatenmodells haben. Eine detaillierte Einführung finden Sie im *ConSol CM Administratorhandbuch (Versionen 6.10 und höher)*, Abschnitt *CM. Resource Pool - Ressourcenrelationen*.

Die folgenden Java-Klassen sind für die Arbeit mit Ressourcen wichtig. Sie stellen allerdings nur einen kleinen Teil der Java-Klassen, die den Ressourcenpool bilden, dar. Eine umfassende Beschreibung aller Klassen und Methoden finden Sie in der ConSol CM Java API Doc.

Objekt	Java-Klasse	Admin-Tool-Beschreibung	Erklärung
`resource`	`Resource`	Ressource	Ein Objekt des definierten Ressourcentyps.
`resourceRelationService`	`Interface ResourceRelationService`	-	Der Service, der viele hilfreiche Methoden für die Behandlung von Ressourcen liefert
`ResourceRelationxy`	`ResourceRelation`	Relation der Ressource/des Ressourcentyps	Relation zwischen einer Ressource und einem anderen Objekt
	`ResourceRelationWithTargetResourceCriteria` `ResourceRelationWithTargetTicketCriteria` `ResourceRelationWithTargetUnitCriteria`	-	Speichert die Kriterien, die die Grundlage für eine Suche bilden, die mit dem `resourceRelationService` durchgeführt wird.

D.12.3.2 Beispiel: Berechnen der Reaktionszeit eines Tickets

Eine Firma kann eine Relation zu einer Ressource des Typs *SLA* haben. In dieser Ressource werden SLA-Daten gespeichert, z. B. die Reaktionszeit. Die gewünschte Deadline für ein Service-Ticket wird berechnet, indem der Wert der Reaktionszeit aus dem Ressourcenobjekt abgerufen wird. Anhand dieses Wertes wird die Deadline für ein Service-Ticket berechnet und das Ergebnis wird in das Datenfeld *Gewünschte Deadline* des Tickets eingetragen.

```
 ///Take SLA (resource) from company of main customer and calculate deadline from
 it
def maincust = ticket.mainContact
def unit_type = maincust.definition.type
println 'TYPE is now : ' + unit_type
def comp
if(unit_type.equals('CONTACT')) {
   comp = maincust.get("company()")
} else if (unit_type.equals('COMPANY')){
   comp = maincust
}

// find SLA relation from SLA resource to the main customer (if company) or the
 company of the main customer (if contact)
def crit = new ResourceRelationWithTargetUnitCriteria()
crit.setUnit(comp)
crit.setResourceTypeName("SLAs")
List<ResourceRelationWithTargetUnit> myrelations =
 resourceRelationService.getByCriteria(crit)
println 'myrelations size is now ' + myrelations.size()
if (myrelations.size() > 0) {
   // one unit can have only one SLA as relation, see AT definition
   def my_sla = myrelations[0].getSourceResource()
   def sla_name = my_sla.get("SLA_Fields_basic.SLA_Name")
   println 'SLA name is now ' + sla_name
   def react_days = my_sla.get("SLA_Fields_basic.ReactionTime")
   // calculate reaction time
   def now = new Date()
   def deadline = now + 1
   ticket.set("serviceDesk_fields.desiredDeadline",deadline)
}
```

Code-Beispiel 79: *Berechnen der Ticket-Deadline aus der SLA. SLA als Ressource, die mit dem Ticket verknüpft ist.*

D.13 Arbeiten mit Textklassen

In diesem Kapitel werden folgende Themen behandelt:

D.13.1 Einleitung

Eine **Textklasse** ist eine Klassifizierung, die Sie einem Ticketeintrag zuweisen. Dies kann sein:

- ein Kommentar
- eine E-Mail, die aus dem Ticket gesendet wurde
- eine E-Mail, die vom Ticket erhalten wurde
- ein Attachment

Die Zuweisung von Textklassen kann mehreren Zwecken dienen:

- Hervorheben des Texts im Ticket in einer bestimmten Farbe, damit er leichter wiedergefunden wird (z. B. ein wichtiger Hinweis wie in der folgenden Abbildung gezeigt). Zusätzlich kann für jede Textklasse ein Icon verwendet werden.
- Kennzeichnen eines Ticketeintrags, damit er in CM.Track sichtbar ist, d. h. damit er für die Kunden verfügbar ist, die sich im ConSol CM-Kundenportal anmelden.
- Kennzeichnen eines Eintrags zur Steuerung des Prozessablaufs, z. B. kann ein Ticket nur abgeschlossen werden, wenn mindestens ein Eintrag als *Lösung* markiert ist.
- Kennzeichnen eines Eintrags für die Übergabe an einen anderen Prozess, z. B. werden die Einträge, die als *Frage* und *Antwort* gekennzeichnet sind, automatisch für ein FAQ-Ticket verwendet.

Auf diese Weise können Sie mit Textklassen die Informationen innerhalb des Tickets organisieren und den Prozessablauf sowie die Verfügbarkeit von Informationen steuern.

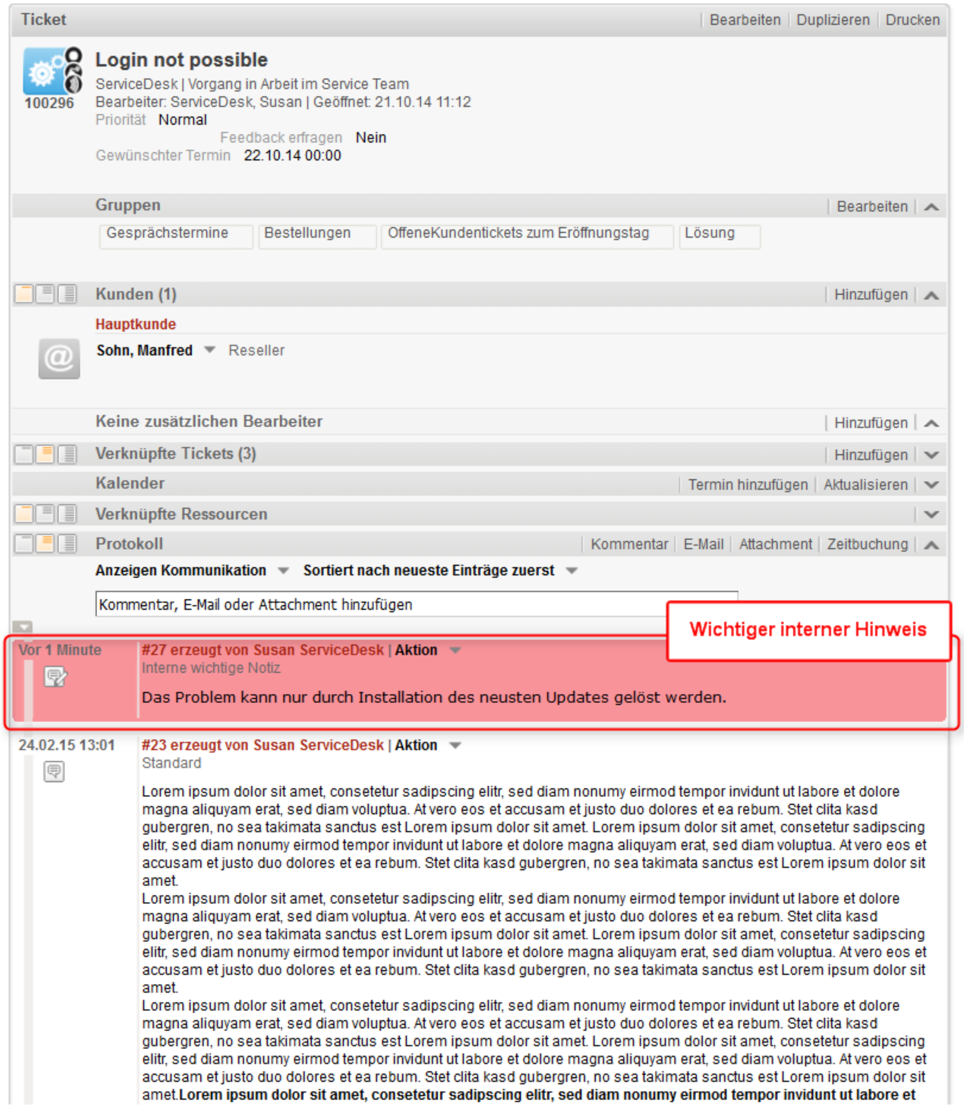


Abbildung 153: *ConSol CM Web Client - Verwenden einer Textklasse für einen wichtigen internen Hinweis*

Eine detaillierte Beschreibung über die Konfiguration und Verwaltung von Textklassen finden Sie im *ConSol CM Administratorhandbuch*. Die Arbeit mit Textklassen im Web Client ist im *ConSol CM Benutzerhandbuch* erklärt.

Die folgenden Code-Abschnitte zeigen Beispiele aus Anwendungsfällen, die häufig bei der täglichen Arbeit der Consultants auftreten.

D.13.2 Beispiel: Überprüfen, ob eine Lösung vorhanden ist, bevor das Ticket geschlossen werden kann

Im folgenden Beispiel werden alle Texteinträge eines Tickets (d. h. Kommentare und E-Mails) überprüft. Wenn die Textklasse *Lösung* mindestens einmal gesetzt ist (dies könnte noch genauer überprüft werden, um zu schauen, ob es genau einen Eintrag *Lösung* gibt), kann das Ticket geschlossen

werden, d. h. das Skript wird fortgesetzt.

Wenn die Textklasse *Lösung* noch nicht gesetzt wurde, wird im Web Client eine Fehlermeldung angezeigt. Wir arbeiten hier mit selbstdefinierten Bezeichnungen. Eine detaillierte Erklärung finden Sie im *ConSol CM Administratorhandbuch*, Abschnitt *Bezeichnungen*. Auf diese Weise ist es nicht erforderlich, die Meldungen für die unterschiedlichen Sprachen hier im Skript zu definieren. Stattdessen erfolgt dies automatisch, da die Bezeichnungen lokalisiert wurden.

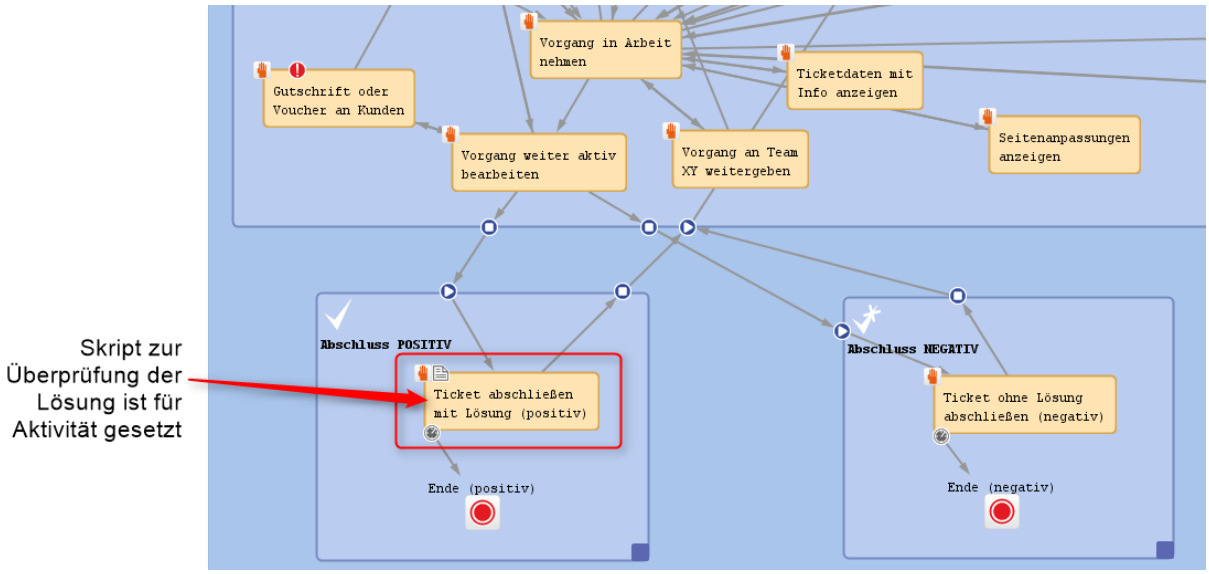


Abbildung 154: *Workflow-Aktivität, in der das Skript verwendet wird*

```
// ticket can only be positively closed when solution - see class of text - is
 provided
def sol_ok = false
List<TextEntry> te_list = workflowApi.ticketText

te_list.each() { te ->
   def cont_class = te.contentEntryClass?.name
   if(cont_class.equals('Solution')){
      sol_ok = true
   }
}

if (!sol_ok) {
   def mylocale = engineerService.getCurrentLocale()
   text = messageProviderService.getMessage("error.solution", mylocale)
   workflowApi.addValidationError("INFO", text)
}
```

Code-Beispiel 80: *Workflow-Skript zum Überprüfen, ob im Ticket eine Lösung definiert wurde*

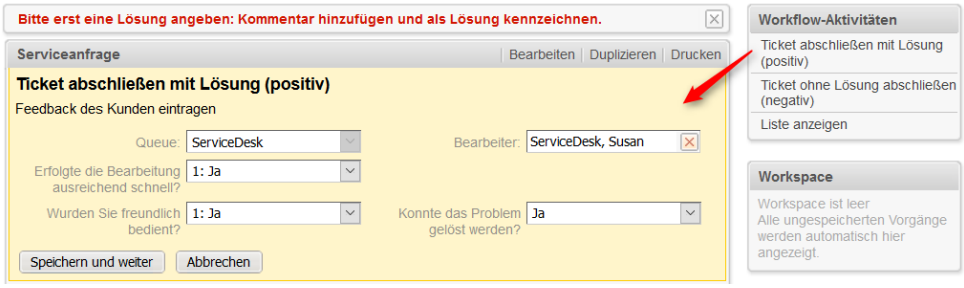


Abbildung 155: *Web Client: Fehlermeldung, wenn im Ticket keine Lösung definiert wurde*

D.13.3 Beispiel: Hinzufügen eines Texts als Ticketkommentar und Setzen einer Textklasse

Das folgende Beispiel zeigt ein postActivityExecutionScript. Wenn die Aktivität *Ticket abschließen mit Lösung (positiv)* im ServiceDesk-Workflow aufgerufen wird, wird automatisch ein FAQ-Ticket erstellt. Der Text, der im ServiceDesk-Ticket als *Lösung* markiert wurde (siehe vorheriges Beispiel) wird als Kommentar in das neue FAQ-Ticket geschrieben und wiederum mit der Textklasse *Lösung* gekennzeichnet. Auf diese Weise wird ein neuer FAQ-Vorschlag mit einer Lösung erstellt, die bewiesenermaßen funktioniert.

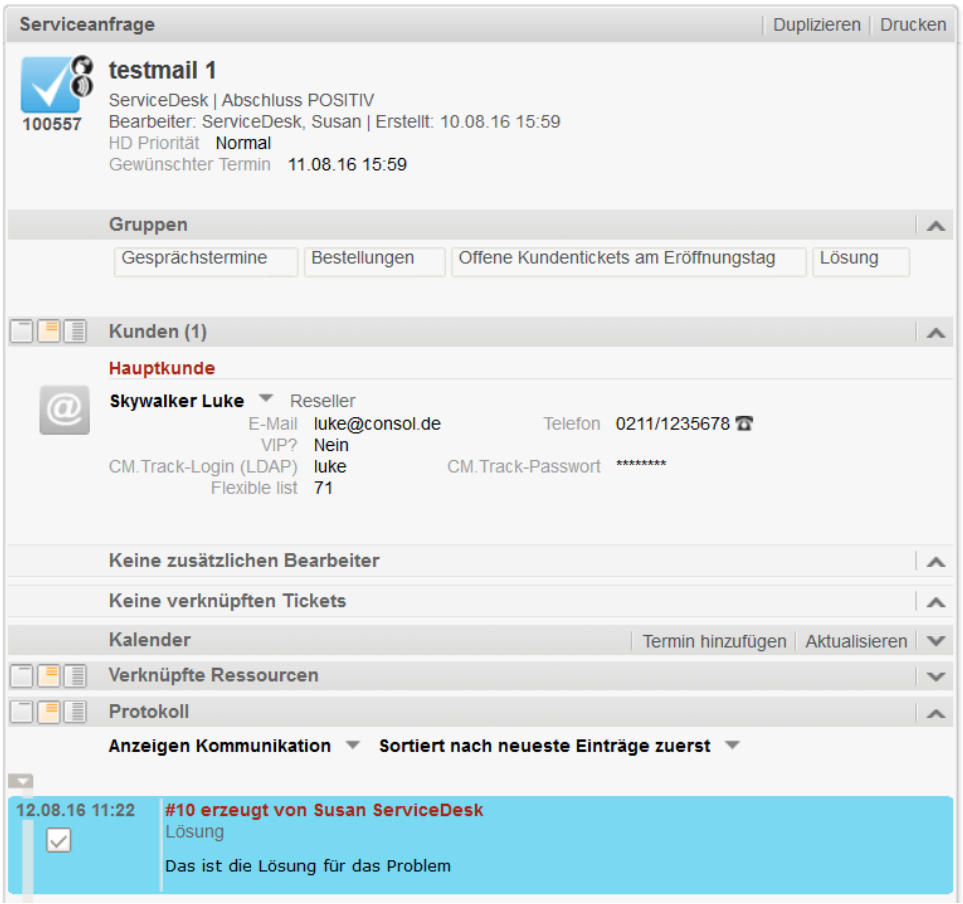
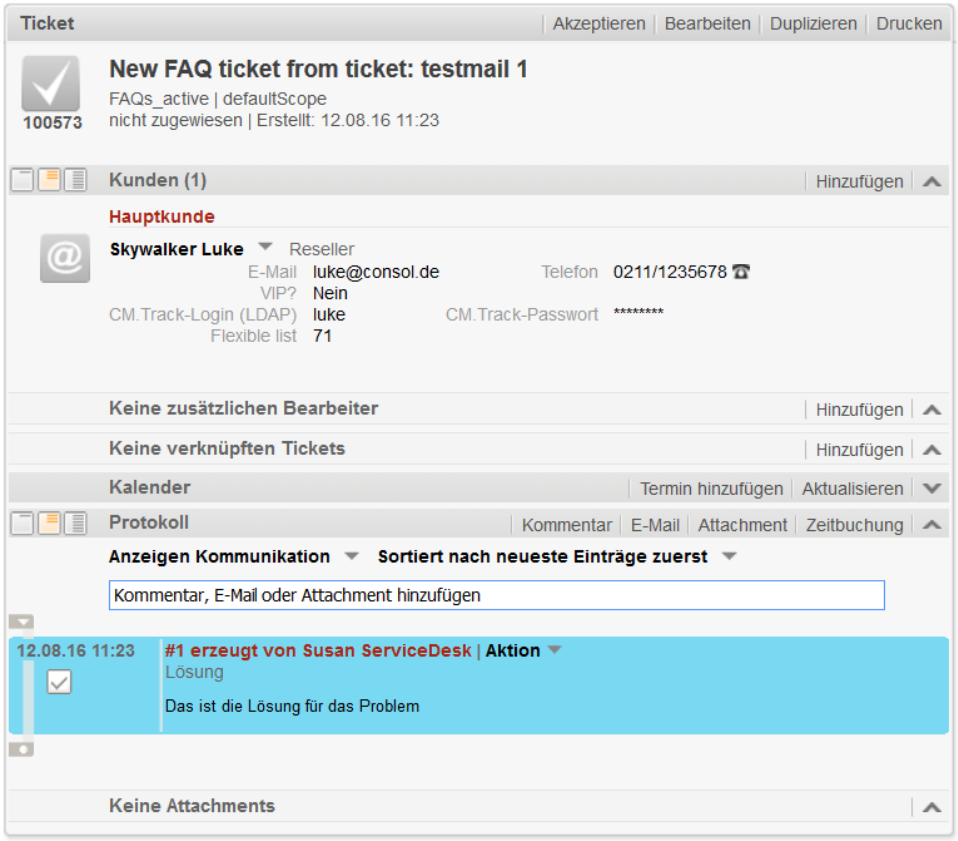


Abbildung 156: *ServiceDesk-Ticket mit Lösung*



Code-Beispiel 81: *Neues FAQ-Ticket mit Lösungstext*

```
import com.consol.cmas.common.model.ticket.Ticket
import com.consol.cmas.common.model.customfield.Unit
import com.consol.cmas.core.server.service.action.PostActionType
import com.consol.cmas.common.model.content.TextEntry
import com.consol.cmas.common.model.content.ContentEntry
switch(activity.name){
   // other cases ...

   case 'defaultScope/Service_Desk/End_positive/Close_ticket_with_solution': M:{
      Ticket newtic = new Ticket()
      def faq_queue = queueService.getByName("FAQs_active")
      newtic.setQueue(faq_queue)
      Unit mycont = ticket.getMainContact()
      newtic.setSubject("New FAQ ticket from ticket: " + ticket.getSubject())
      ticketService.createWithUnit(newtic,mycont)
      // add solution text from parent ticket as ticket comment
      List<ContentEntry> ce_list = ticketContentService.getContentEntries
       (ticket,TextEntry.class)

      ce_list.each() { ce ->
         def cont_class = ce.contentEntryClass?.name
         if(cont_class.equals('Solution')){
            def mytext = ce.text
            def new_te = new TextEntry("Solution from SD Ticket", mytext)
            def ce_class = ce.contentEntryClass
            new_te.setContentEntryClass(ce_class)
            ticketContentService.createContentEntry(newtic, new_te)
         }
      }
   }
}
```

Code-Beispiel 82: *Auszug aus dem postActivityExecutionScript*

Beachten Sie, dass es für den Bearbeiter in diesem Beispiel nicht offensichtlich ist, dass "hinter den Kulissen" ein FAQ-Ticket erstellt wird. Dies ist in besonderen Fällen möglicherweise erforderlich. Wenn Sie eine solche Aktion "hinter den Kulissen" nicht möchten, können Sie den gesamten Code in die Workflow-Aktivitäten schreiben. Wenn Sie den Skript-Code direkt in das Skript der Workflow-Aktivität schreiben, können Sie Methoden wie *workflowApi.getTicketText()* und *workflowApi.addTicketText()* verwenden.

Wenn Sie den Code in ein Admin-Tool-Skript schreiben und dieses Skript aus einem Workflow-Skript aufrufen, müssen Sie die im obigen Beispiel gezeigten Methoden von *TicketContentService* verwenden.

D.14 Arbeiten mit Attachments

In diesem Kapitel werden folgende Themen behandelt:

D.14.1 Einleitung

In ConSol CM können Sie Attachments an Tickets anhängen. Diese Attachments können verschiedene Dateitypen haben und direkt aus dem Ticketprotokoll geöffnet werden, sofern die richtige Applikation auf dem Client-Rechner installiert ist. Eine detaillierte Erklärung der Arbeit mit Attachments finden Sie im *ConSol CM Benutzerhandbuch*.

Attachments können in unterschiedlichen Prozessschritten in CM integriert werden:

- Es wird eine E-Mail mit Attachment an ConSol CM gesendet.
  - Die E-Mail wird an ein vorhandenes Ticket angehängt. Das Attachment wird an das vorhandene Ticket angehängt.
  - Es wird ein neues Ticket erstellt. Das Attachment wird an das neue Ticket angehängt.
- Ein Bearbeiter hängt über den Ticketeditor eine Datei an das Ticket.
- Ein Bearbeiter verwendet CM.Doc und ein Attachment (Microsoft Word oder OpenOffice) wird automatisch erstellt und an das Ticket angehängt.

Alle diese Attachments sind letztendlich Ticket-Attachments mit unterschiedlichen Dateitypen. Die Groovy-Klasse, die für die entsprechenden Objekte verwendet wird, ist `com.consol.cmas.common.model.content.AttachmentEntry`.

Bitte beachten Sie Folgendes:

... `workflowApi.workflowApi.getAttachmentList()` gibt nur die Attachments zurück, die direkt an das Ticket angehängt wurden, d. h. die manuell oder automatisch an das Ticket angehängt wurden. E-Mail-Attachments gehören nicht dazu!

... `ticketContentService.getAttachmentEntries(ticket, ContentEntryCategory.values())` gibt alle Ticket-Attachments zurück.

... `ticketContentService.getAttachmentEntries(ticket, ContentEntryCategory.INCOMING_MAIL)` gibt alle Ticket-Attachments zurück, die aus E-Mail-Attachments in eingehenden E-Mails hervorgegangen sind. Andere mögliche Werte sind OUTGOING_MAIL oder DEFAULT.

D.14.2 Beispiel 1: Anhängen aller Attachments eines ServiceDesk-Tickets an das Child-Ticket

Der folgende Code stammt aus einem Admin-Tool-Skript, das aus der Workflow-Aktivität *Aufgabe erstellen mit Übergabe Produktliste* aufgerufen wird. Aus dem ServiceDesk-Ticket wird ein neues Ticket in der Queue *Aufgaben* erstellt. Alle Attachments (oder nur die wichtigen, siehe alternative

Lösung) werden an das neue Child-Ticket übertragen. Die Produktliste wird ebenfalls übertragen.

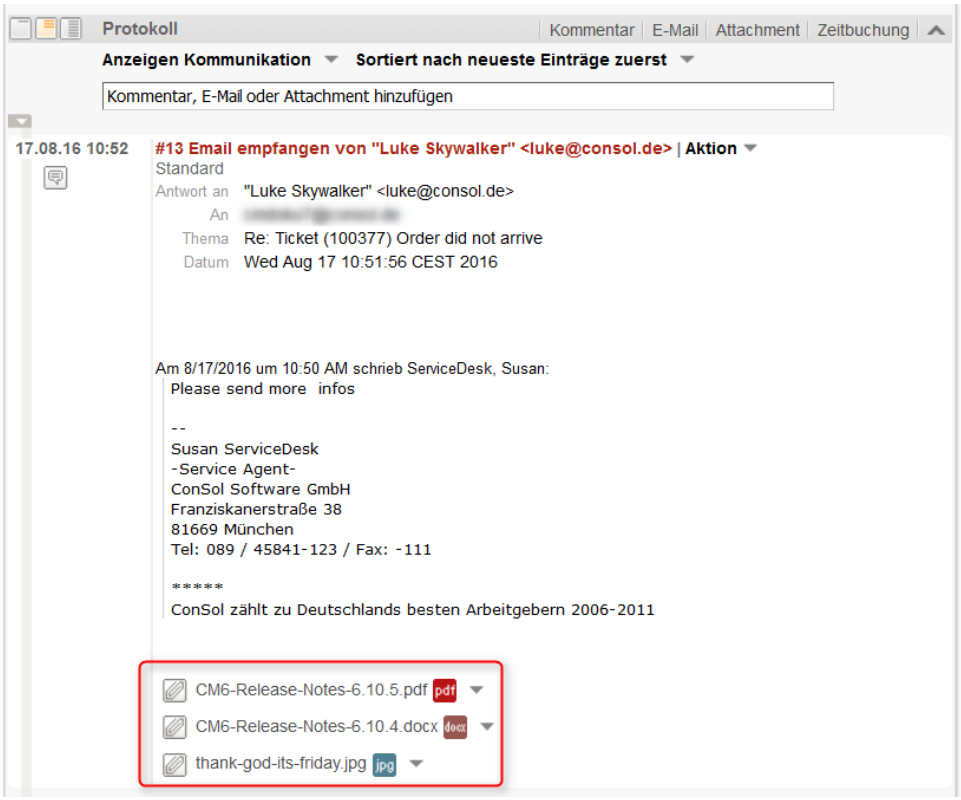


Abbildung 157: *Web Client: ServiceDesk-Ticket mit drei Attachments*

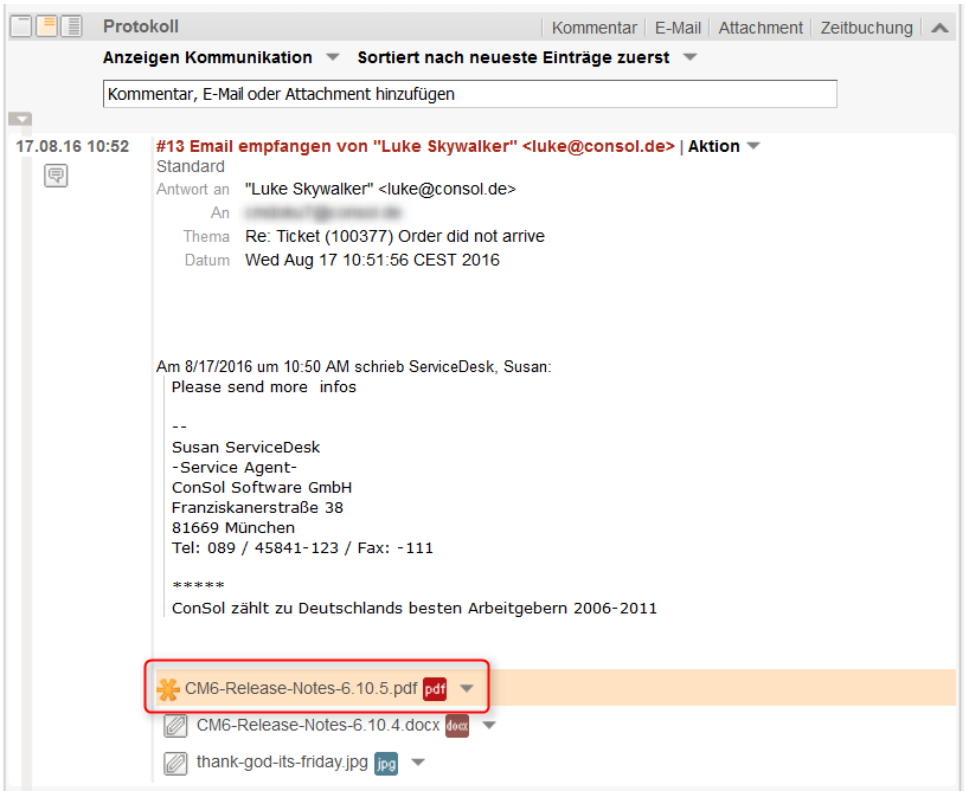


Abbildung 158: *Web Client: ServiceDesk-Tickets mit drei Attachments, eines ist über die entsprechende Textklasse als wichtiges Attachment gekennzeichnet*

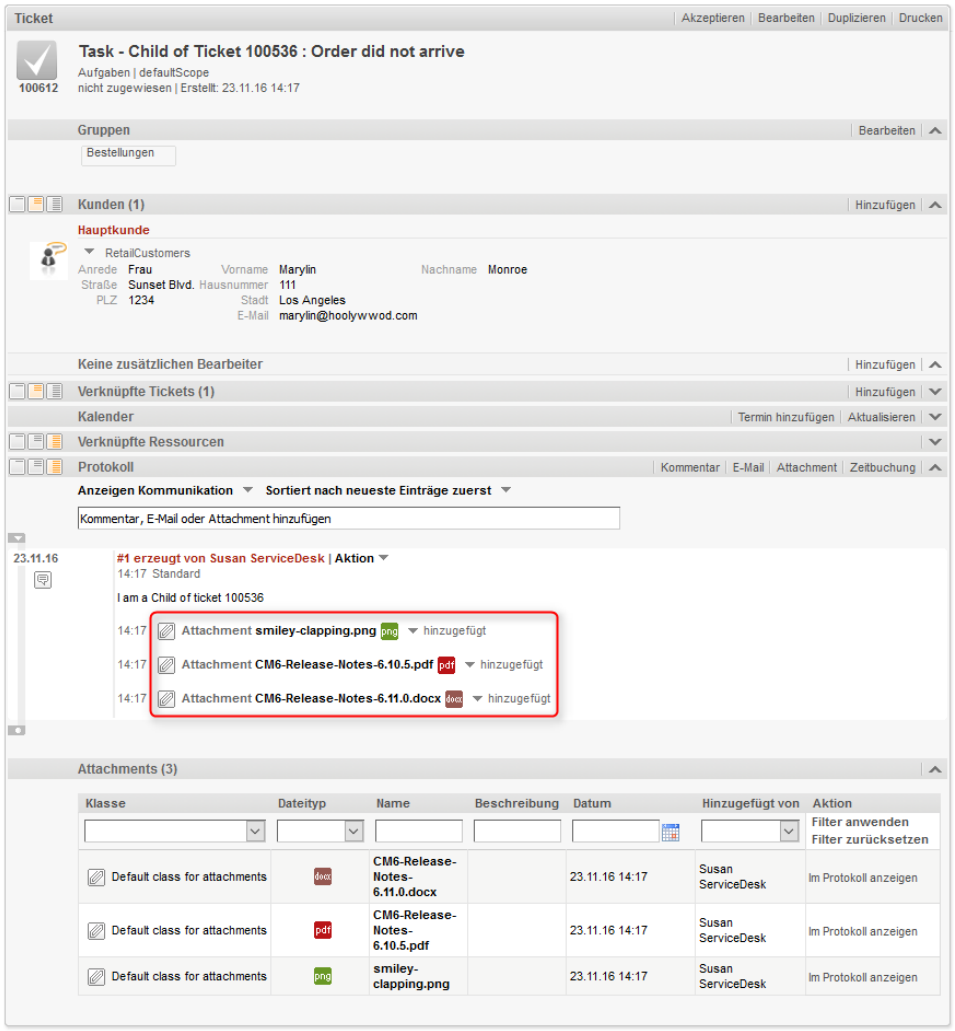


Abbildung 159: *Web Client: Child-Ticket mit drei Attachments*

```
import com.consol.cmas.common.model.ticket.Ticket
import com.consol.cmas.common.service.*
import com.consol.cmas.common.model.content.AttachmentEntry
import com.consol.cmas.common.model.customfield.cfel.Struct
import com.consol.cmas.common.model.content.ContentEntryCategory
import com.consol.cmas.common.model.content.ContentFile
Ticket ticket = workflowApi.ticket
Ticket nt = new Ticket()
def qu = queueService.getByName("SpecialTasks")
nt.setQueue(qu)
def subj = "Task - Child of Ticket " + ticket.getId() + " : " + ticket.getSubject()
nt.setSubject(subj)
nt.setEngineer(null)
// main contact:
// def cont = ticket.getMainContact()
def cont = workflowApi.getPrimaryContact()
def text = "I am a Child of ticket " + ticket.getId()
def orders = ticket.get("order_data.orders_list")?.each() { ord ->
   nt.add("order_data.orders_list", new Struct().set("orders_hardware", ord.orders_
    hardware.getName())
   .set("orders_contact", ord.orders_contact)
   .set("orders_number", ord.orders_number)
   )
}

// put copy of each attachment to each child ticket
workflowApi.createChildTicket(nt, text,cont)

List<AttachmentEntry> attachmnts = ticketContentService.getAttachmentEntries
 (ticket, ContentEntryCategory.values())

attachmnts.each(){ at ->
   if ( at.file ) { // ignores deleted attachments
      def new_at = new AttachmentEntry()
      new_at.mimeType = at.file.mimeType ?: at.mimeType
      new_at.file = new ContentFile(at.file.name,new_
       at.mimeType,at.file.inputStream,at.file.size)
      new_at.description = at.description
      workflowApi.addAttachment(nt,new_at)
   }
}
```

Code-Beispiel 83: *Admin-Tool-Skript, das aus einer Workflow-Aktivität aufgerufen wird: Child-Ticket erstellen und die Produktliste und alle Ticket-Attachments übertragen*

Seit CM-Version 6.9.2.0 kann ein Attachment eine Textklasse haben. Wenn Sie nur die Attachments mit der Textklasse *Wichtiges Attachment* übertragen möchten, können Sie in jeder Schleife folgenden Code verwenden:

```
attachmnts.each(){ at ->
   if ( at.file && at.contentEntryClass?.name?.equals("Important attachment")) { //
    ignores deleted attachments
      def new_at = new AttachmentEntry()
      new_at.mimeType = at.file.mimeType ?: at.mimeType
      new_at.file = new ContentFile(at.file.name,new_
       at.mimeType,at.file.inputStream,at.file.size)
      new_at.description = at.description
      workflowApi.addAttachment(nt,new_at)
   }
}
```

Code-Beispiel 84: *Alternative (zusätzliche) Lösung: Nur die wichtigen Attachments übertragen*

D.15 Suchen nach Tickets, Kunden und Ressourcen über die ConSol CM-Workflow-API

In diesem Kapitel werden folgende Themen behandelt:

D.15.1 Einleitung

In ConSol CM können Sie die Datenbank nach Tickets oder Units (Kontakten und Firmen) durchsuchen. Wenn Ihr CM-System das Modul CM.ResourcePool enthält, können Sie die Datenbank auch nach Ressourcen durchsuchen. Alle Suchmodi basieren auf dem gleichen Prinzip:

1. Es wird ein Objekt *criteria* erstellt, in dem alle Parameter für die Zielobjekte gespeichert werden.
    a. **TicketCriteria** für Tickets
    b. **UnitCriteria** für Kontakte und Firmen
    c. **ResourceCriteria** für Ressourcen
2. Das Criteria-Objekt wird an einen Service übergeben, der dann eine Liste mit den Ergebnisobjekten zurückgibt.
    a. **TicketService** für Tickets
    b. **UnitService** für Units
    c. **ResourceService** für Ressourcen

Die Felder, die als Parameter für die Criteria-Objekte gesetzt sind, müssen indiziert sein, d. h. die Annotationen *field-indexed* muss gesetzt sein.

D.15.2 Suchen nach Tickets

Um nach Tickets zu suchen, müssen Sie das Objekt *TicketCriteria* erstellen. Es können zum Beispiel folgende Felder gesetzt werden (siehe auch entsprechende *setter-Methoden* in der folgenden Abbildung):

- Datum der Ticketerstellung
- Bearbeiter
- Systemspezifische Benutzerdefinierte Felder
- Kriterien des Ticketprotokolls
- Ticket-IDs
- Änderungsdatum
- Ticketname
- Muster für den Ticketbetreff
- Queue-IDs
- IDs für die aktuellen Workflow-Bereiche
- Aktueller Status (geschlossen/offen)
- Zusätzliche Bearbeiter

void	setCreationDateRange(DateRange pCreationDateRange)
void	setEngineerCriteria(TicketCriteria.EngineerCriteria pEngineerCriteria)
void	setExcludeIds(boolean pExcludeIds)
void	setFields(Set<AbstractField> pFields)
void	setHistoryCriteria(TicketCriteria.HistoryCriteria pHistoryCriteria)
void	setIdRange(org.apache.commons.lang.math.LongRange pIdRange)
void	setIds(Set<Long> pIds)
void	setModificationDateRange(DateRange modificationDateRange)
void	setName(String pName)
void	setPattern(String pPattern)
void	setPermission(QueuePermissionType pPermission) Set permission to filter out the unwanted results (along with READ).
void	setQueueIds(Set<Long> pQueueIds)
void	setResourceRelations(Set<ResourceRelationRelatedElementCriteria<?>> pResourceRelations)
void	setScopeIds(Set<Long> pScopeIds)
void	setStatus(TicketCriteria.Status pStatus)
void	setSubject(String pSubject)
void	setUserCriteria(Set<TicketUserCriteria> pUserCriteria)

Abbildung 160: *Setter-Methoden der Klasse TicketCriteria, Java API Doc, CM-Version 6.10.5.2*

Das Objekt *TicketCriteria* muss an den *TicketService* übergeben werden, der implizit in jedem Skript als Singleton *ticketService* verfügbar ist. Details über die Klassen und Methoden finden Sie in der Dokumentation zur *ConSol CM Workflow Java API*.

D.15.2.1 Beispiel 1: Allgemeines Beispiel für eine Suche nach Tickets

```
def ticketCrit = new TicketCriteria()
ticketCrit.subject = "TICKET_SUBJECT"
ticketCrit.setQueueIds([new Long(workflowApi.getQueueByName("QUEUE_NAME").id)] as
 Set)
ticketCrit.setFields([new StringField(new FieldKey("FIELD_GROUP", "FIELD_NAME"),
 "SEARCH_VALUE")] as Set)
List<Ticket> foundTickets = ticketService getByCriteria(ticketCrit)
def firstTicket = foundTickets?.first()
```

Code-Beispiel 85: *Suche nach Tickets (Pseudocode)*

D.15.2.2 Beispiel 2: Finden aller Tickets mit dem gleichen Modul wie das aktuelle Ticket

Das folgende Beispiel stammt aus einem Workflow einer ServiceDesk-Umgebung. Wenn das Ticket erstellt wurde und das Modul aus einer Liste gesetzt wurde, soll der Workflow automatisch überprüfen, ob es andere offene Tickets mit dem gleichen Modul gibt. Für das Modul wird ein *enum* verwendet.

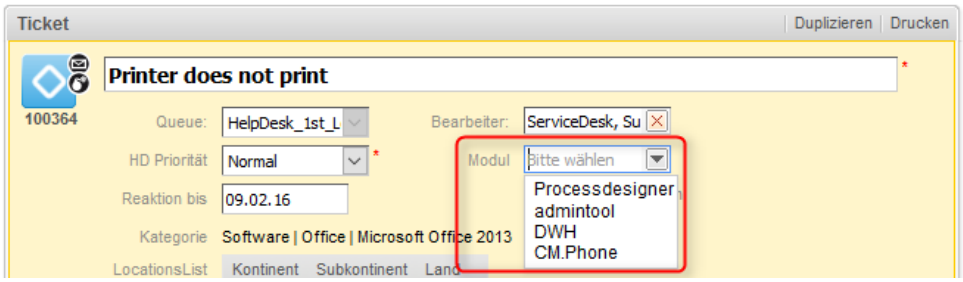


Abbildung 161: *Web Client: Auswählen des Moduls für ein ServiceDesk-Ticket*

```
def mod = ticket.getField("helpdesk_standard", "module")
def crit = new TicketCriteria()
crit.setStatus(TicketCriteria.Status.OPEN)
Set<AbstractField> myfields = [mod]
crit.setFields(myfields)
List<Ticket> tics = ticketService.getByCriteria(crit)
tics.each() { tic ->
   println 'Next Ticket subject is now ' + tic.subject
   // do whatever is required with the tickets
}
```

Code-Beispiel 86: *Suchen nach Tickets mit dem gleichen Modul wie das aktuelle Ticket*

D.15.2.3 Beispiel 3: Suchen nach Tickets nach Unit

In diesem Beispiel suchen wir das *AccountManagement-Ticket* für eine Firma.

```
import com.consol.cmas.common.model.scripting.unit.PostActionType
import com.consol.cmas.common.model.scripting.unit.PostActionParameter
import com.consol.cmas.common.model.customfield.Unit
import com.consol.cmas.common.model.ticket.TicketCriteria
import com.consol.cmas.common.model.customfield.ListField
import com.consol.cmas.common.model.customfield.ContactReferenceField
import com.consol.cmas.common.model.customfield.UnitReferenceSearchField
import com.consol.cmas.common.model.customfield.ContactReferenceSearchField
import com.consol.cmas.common.model.customfield.meta.FieldKey
import com.consol.cmas.common.model.ticket.Ticket
import com.consol.cmas.common.model.ContactTicketRole
import com.consol.cmas.common.model.customfield.StringField
import com.consol.cmas.common.model.scripting.unit.UnitActionScriptResult

//get AM queue for search
def q_id = (workflowApi.getQueueByName("AccountManagement")).id
def q_ids = new HashSet()
q_ids.add(q_id)

//find AM ticket for the company
def crit = new TicketCriteria()
crit.setQueueIds(q_ids)

// Create List Field Key
def contactSearchListFieldKey = new FieldKey("queue_fields","contacts")

// Prepare List Field
def contactsListField = new ListField(contactSearchListFieldKey )

// Create Memberfield Key
def contactSearchFieldKey = new FieldKey("queue_fields","contacts_member")

// Create Unit Memberfield with Unit and Ticket-Main Role
def contactsMember = new ContactReferenceSearchField(contactSearchFieldKey, unit,
 ContactTicketRole.MAIN_ROLE)

// Put Member Field in Unit List Field
contactsListField.addChild(contactsMember)

// Put prepared fields into TicketCriteria
crit.setFields([contactsListField] as Set)

// Search ... and Result
def foundTickets = ticketService.getByCriteria(crit)
println "Found tickets: ${foundTickets}"
if ( foundTickets ) {
   def AM_tic = foundTickets.first()
   def AM_tic_id = AM_tic.id
}
```

Code-Beispiel 87: *Suche nach Tickets nach Unit*

D.15.2.4 Beispiel 4: Suchen nach Tickets nach Bearbeiter, um die Überlastung der Bearbeiter zu vermeiden

In diesem Beispiel kann ein Bearbeiter die Workflow-Aktivität *Neues Ticket (Ticket annehmen)* nur aufrufen, wenn er noch nicht zu viele Tickets hat. Die maximal zulässige Anzahl an Tickets ist in der kundenspezifischen System-Property *custom-servicedesk,engineer.max.open.tickets* gespeichert. Auf diese Weise kann die Anzahl von einem CM-Administrator ohne Beteiligung eines Workflow-Entwicklers geändert werden.

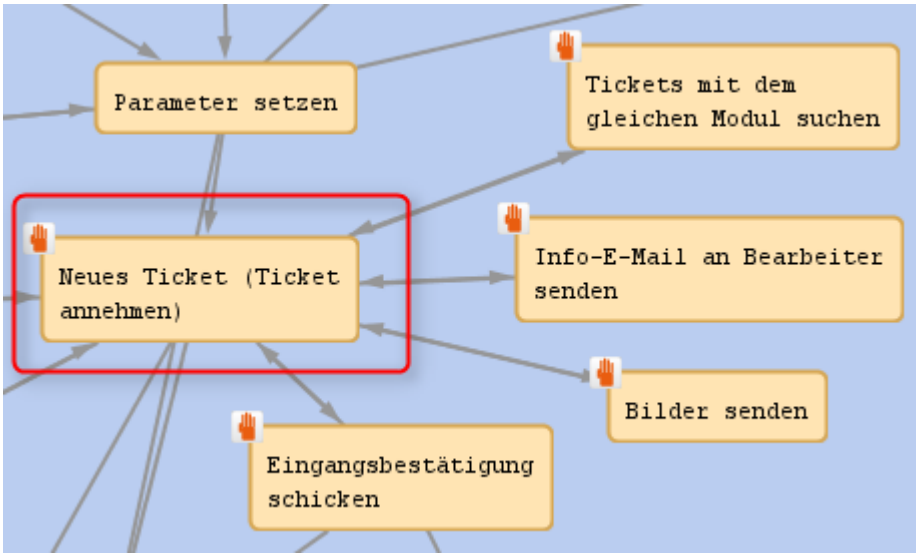


Abbildung 162: *Workflow-Aktivität, die das Skript zur Kontrolle der Ticketanzahl enthält*

```
// Engineer can only accept ticket if he does not have too many tickets already
def curr_eng = workflowApi.currentEngineer
def max_tics = configurationService.getValue("custom-
 servicedesk","engineer.max.open.tickets")
// look for open tickets of current engineer
def engs = []
engs.add(curr_eng.id)
TicketCriteria tic_crit = new TicketCriteria()
tic_crit.engineerCriteria = TicketCriteria.EngineerCriteria.assigned(engs as Set)
tic_crit.status = TicketCriteria.Status.OPEN
List<Ticket> open_eng_tics = ticketService.getByCriteria(tic_crit)
def tic_number = open_eng_tics.size
if (tic_number > max_tics) {

   workflowApi.addValidationError("INFO","You have too many tickets (" + tic_number
    + ") already, so you cannot accept another ticket. Maximum allowed number is "
    + max_tics)

} else {
   ticket.setEngineer(curr_eng)
}
```

Code-Beispiel 88: *Skript der Aktivität "Neues Ticket (Ticket annehmen)"*

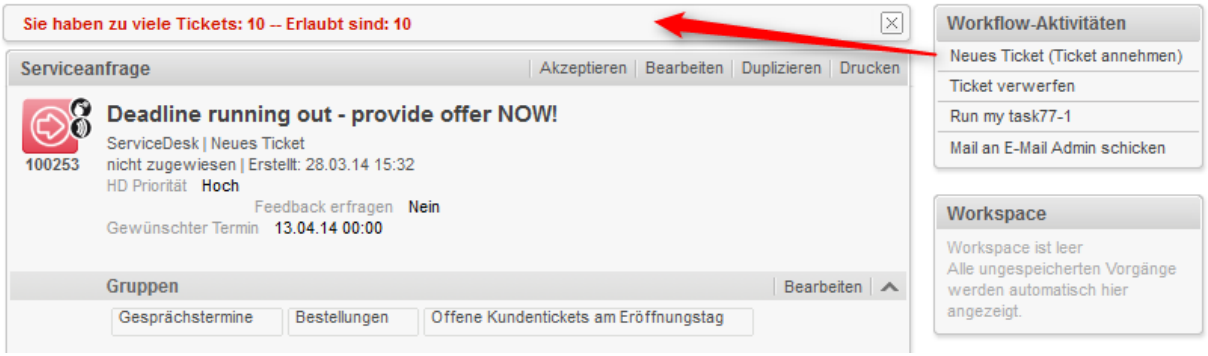


Abbildung 163: *Web Client: Kontrolle der Anzahl der Tickets, die einem Bearbeiter zugewiesen sein können*

D.15.3 Suchen nach Units (Kontakten und Firmen)

Um nach Units (d. h. Kontakten und/oder Firmen) zu suchen, müssen Sie ein Objekt *UnitCriteria* erstellen. Es können zum Beispiel folgende Felder gesetzt werden (siehe auch entsprechende *setter-Methoden* in der folgenden Abbildung):

- Kundengruppe
- Systemspezifische Datenobjektgruppenfelder
- Unit-IDs
- Muster für Units
- Telefonnummer (neu in CM-Version 6.9.3, für CM.Phone verwendet)
- TicketCriteria
- UnitDefinition name
- Boolean UseInCriterion

Verwenden Sie dann *unitService*, um das Suchergebnis abzurufen.

void	setActive(Boolean pActive)
void	setCustomerGroupIds(Set<Long> pCustomerGroupIds)
void	setExcludeIds(boolean pExcludeIds)
void	setFields(Set<AbstractField> pFields)
void	setGroupNames(Set<String> pGroupNames) Deprecated.
void	setIdRange(org.apache.commons.lang.math.LongRange pIdRange)
void	setIds(Set<Long> pIds)
void	setPattern(String pPattern)
void	setPermission(CustomerGroupPermissionType pPermission) Set permission to filter out the unwanted results (along with READ).
void	setPhoneNumber(String pPhoneNumber)
void	setResourceRelations(Set<ResourceRelationRelatedElementCriteria<?>> pResourceRelations)
void	setTicketCriteria(Set<AbstractField> pCallUnitReferences, TicketCriteria pTicketCriteria)
void	setUnitDefinitionNames(Set<String> pUnitDefinitionNames)
void	setUseInCriterion(boolean pUseInCriterion)

Abbildung 164: *Setter-Methoden der Klasse UnitCriteria, Java API Doc, CM-Version 6.10.5.2*

D.15.3.1 Beispiel 1: Suchen nach Kontakten nach dem Vornamen und Nachnamen

```
def unitCrit = new UnitCriteria()
unitCrit.setFields([new StringField(new FieldKey("UNIT_GROUP_NAME", "firstname"),
 "Max"),
new StringField(new FieldKey("UNIT_GROUP_NAME", "lastname"), "Mustermann")] as Set)
def foundContacts = unitService.getByCriteria(unitCrit)
def firstContact = foundContacts?.first()
```

Code-Beispiel 89: *Suche nach Kontakten nach Vornamen und Nachnamen*

D.15.3.2 Allgemeine Syntax für die Unit-Suche nach einem Enum-Wert

```
import com.consol.cmas.common.model.customfield.UnitCriteria
import com.consol.cmas.common.model.customfield.EnumSearchField
import com.consol.cmas.common.model.customfield.meta.FieldKey

def unitCrit = new UnitCriteria()
def companyEnumField = new EnumSearchField(new FieldKey("customer", "company"),
 [enumService.getValueByName("ENUM_GROUP_NAME",ENUM_VALUE_NAME)] as Set)
unitCrit.setFields([companyEnumField] as Set)
unitService.getByCriteria(unitCrit).each { foundContact ->
   println "Processing found contact: "+foundContact.get("name")
}
```

Code-Beispiel 90: *Suche nach Units nach Enum-Wert (allgemeine Syntax)*

D.15.3.3 Beispiel 2: Suchen nach Units nach Enum-Wert

```
def unitCrit = new UnitCriteria()

//all other UnitCriteria init operations skipped

// this is the requested value inside the list:
def secLvl = ticket.get("transportEntryData.securityLevel")

//ShipperData/securityLevel is the path of the EnumField inside the list
def secLvlEnumFieldKey = new FieldKey("ShipperData","securityLevel")

//create the template field with FieldKey and our value to search for
def secLvlTemplateField = new EnumField(secLvlEnumFieldKey, secLvl)

//ShipperData/securityLevels is the path of the list itself
def secLvlListTemplateFieldKey = new FieldKey("ShipperData","securityLevels")

//init the template list with the value to be searched for
def secLvlListTemplateField = new ListField(secLvlListTemplateFieldKey,
 [secLvlTemplateField])

// put the template list into the UnitCriteria object
def unitCrit.setFields([secLvlListTemplateField] as Set)

// Search ... and Result
def shippers = unitService.getByCriteria(unitCrit)
```

Code-Beispiel 91: *Suche nach Units nach Enum-Wert (Beispiel)*

D.15.4 Suchen nach Ressourcen

Um nach Ressourcen zu suchen, müssen Sie das Objekt *ResourceCriteria* erstellen. Es können zum Beispiel folgende Felder gesetzt werden (siehe auch entsprechende *setter-Methoden* in der folgenden Abbildung):

- Datumsbereich
- Ausgeschlossene Ressourcen-IDs
- Ressourcengruppen-IDs der gewünschten Ressourcen

Verwenden Sie dann *resourceService*, um das Suchergebnis abzurufen.

void	setAccessModeDateRange(DateRange pAccessModeDateRange)
void	setActive(Boolean pActive)
void	setExcludeIds(boolean pExcludeIds)
void	setFields(Set<AbstractField> pFields)
void	setIdRange(org.apache.commons.lang.math.LongRange pIdRange)
void	setIds(Set<Long> pIds)
void	setModificationDateRange(DateRange pModificationDateRange)
void	setPattern(String pPattern)
void	setPermission(ResourceTypePermissionType pPermission) Set permission to filter out the unwanted results (along with READ).
void	setResourceGroupsIds(Set<Long> pResourceGroupsIds)
void	setResourceGroupsTechnicalNames(Set<String> pResourceGroupsTechnicalNames)
void	setResourceRelations(Set<ResourceRelationRelatedElementCriteria<?>> pResourceRelations)
void	setResourceTypesIds(Set<Long> pResourceTypesIds)
void	setResourceTypesTechnicalNames(Set<String> pResourceTypesTechnicalNames)

Abbildung 165: *Setter-Methoden der Klasse ResourceCriteria, Java API Doc, CM-Version 6.10.5.2*

D.15.4.1 Erstellen einer Liste des IT-Inventars und Schreiben eines Kommentars in ein Ticket

Im folgenden Beispiel wird eine Liste aller IT-Assets, die als Ressourcen eines bestimmten Ressourcentyps abgebildet sind, in das aktuelle Ticket geschrieben.

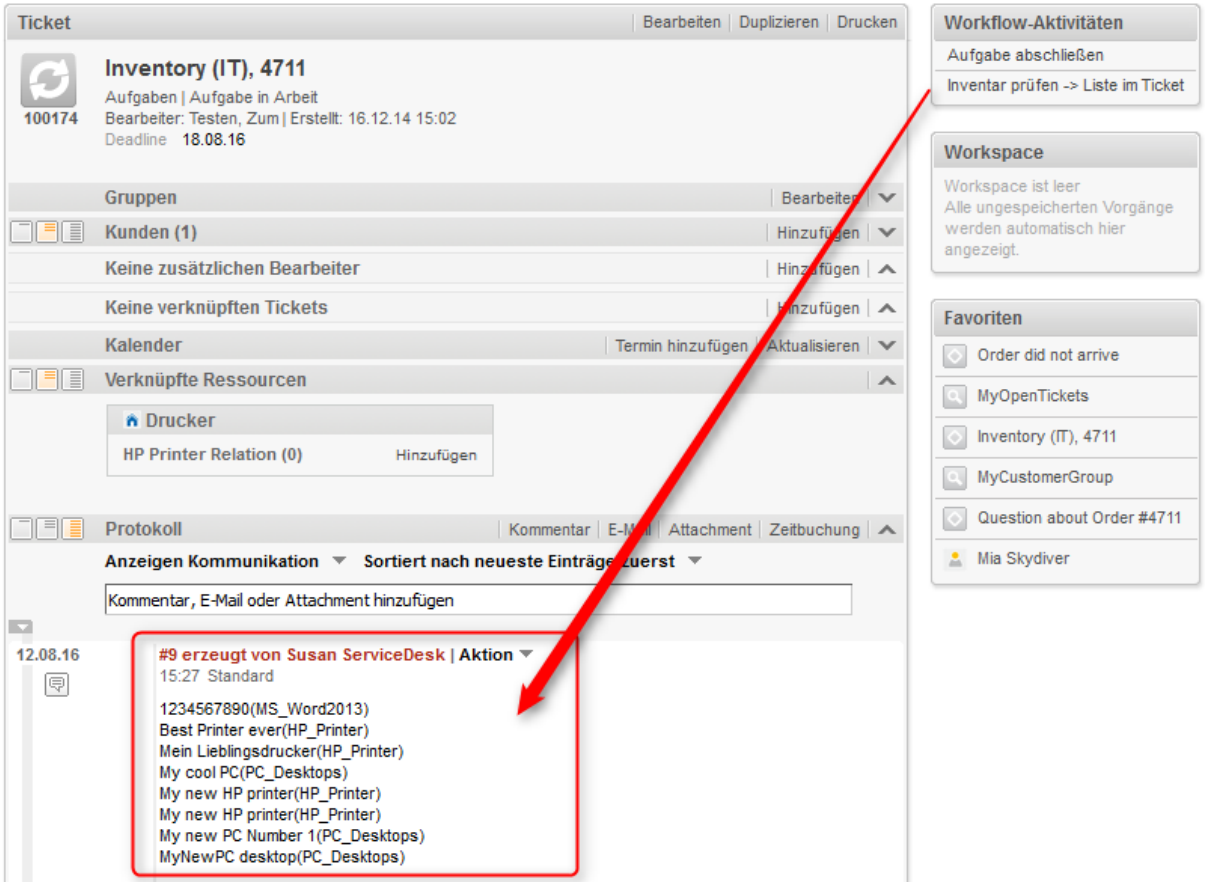


Abbildung 166: *Web Client: Aufrufen einer Workflow-Aktivität, die die Liste der IT-Assets in einen Kommentar im Ticket schreibt (Skript: siehe folgendes Code-Beispiel)*

```
// Make an inventory of the asset base (CM.Resource Pool) and write the info into
 the current ticket
// Asset base contains Hardware and Software resources
import com.consol.cmas.common.model.resource.*
import java.util.Arrays
import com.consol.cmas.common.model.ticket.Ticket
import com.consol.cmas.common.model.content.TextEntry
ticket = workflowApi.getTicket()
def crit = new ResourceCriteria()
// only three resource groups are required:
Set<String> res_groups = ["Printers","OfficeSoftware","PCs"]
crit.setResourceGroupsTechnicalNames(res_groups)
List<Resource> res_list = resourceService.getByCriteria(crit)
def desc_name def printout_list = []
res_list.each(){res ->
   def res_tname = res.getResourceType().getName()
   // println 'Resource type is ' + res_tname
   // find the field which is used for description of a single resource
   // this depends on the resource fields of the resource field group
   switch (res_tname){
      case "HP_Printer": desc_name = "HP_Printer_Fields_basic.name"
      break;
      case "MS_Word2013": desc_name = "MS_Word2013_Fields.OrderNumber"
      break;
      case "PC_Desktops": desc_name = "PC_Desktop_Fields_basic.name"
      break;
      case "PC_Laptops": desc_name = "PC_Laptop_Fields_basic.laptopname"
      break;
   }

   def res_final = res.get(desc_name) + '(' + res_tname + ')<br>'
   printout_list += res_final
}
printout_list = printout_list.sort{it}

TextEntry new_te = new TextEntry("Asset Base contains the following
 assets",printout_list.toString()
.replace('[','')
.replace(']','')
.replace(',',''))

ticketContentService.createContentEntry(ticket,new_te)
```

Code-Beispiel 92: *Schreiben einer Liste mit IT-Assets (Ressourcen) ins Ticket*

D.16 Debug-Informationen

In diesem Kapitel werden folgende Themen behandelt:

D.16.1 Einleitung

Manchmal möchten Sie unter Umständen die Ausgabe eines Workflow- oder Admin-Tool-Skripts mithilfe der Debug-Ausgabe in den Log-Dateien überprüfen. In ConSol CM wird die Debug-Ausgabe normalerweise in die Datei *server.log* geschrieben, die sich (in der Standardkonfiguration) an folgendem Pfad befindet:

- **In JBoss 5:**
  <JBOSS_HOME>\log\server.log
- **In JBoss 7 (einzelne Instanz):**
  <JBOSS_HOME>/standalone/server.log
- **In Oracle WebLogic:**
  <DOMAIN_HOME>\cm-logs und
  <DOMAIN_HOME>\cmrf-logs\server.log

Die Logging-Konfiguration kann geändert werden, indem Sie die entsprechende Konfigurationsdatei editieren:

- In JBoss 5:
  <JBOSS_HOME>/conf/ jboss-log4j.xml
- In JBoss 7:
  <JBOSS_HOME>/standalone/configuration/cm6.xml oder cm6-cmrf.xml
- In Weblogic:
  <WLS_HOME>user_projects\domains\consolcm6_domain\log4j.xml.

Eine umfassende Dokumentation des Logging und der Log-Dateien in ConSol CM finden Sie im ConSol CM Set-Up Manual.

Als Alternative können Sie die Informationen auch als Text ins Ticket schreiben.

D.16.2 Verwenden von Anweisungen für die Debug-Ausgabe

D.16.2.1 Debug-Ausgabe in der Datei server.log

Mit den folgenden Anweisungen können Sie Log-Informationen in die Datei *server.log* schreiben. Dies funktioniert in Workflow-Skripten und in Admin-Tool-Skripten.

```
println 'This is my debug message.'
```

```
println("This is my debug message.")
```

```
log.info("This is my debug message.")
```

```
log.info "This is my debug message."
```

⚠ In einem WebLogic-System muss normalerweise die Anweisung *log.info* verwendet werden. Unter Umständen funktioniert *println* nicht.

D.16.2.2 Debug-Ausgabe als Texteintrag im Ticket

Wenn Sie die Informationen im Ticket anzeigen möchten (z. B. weil Sie keinen Zugriff auf das Dateisystem, auf dem die Log-Dateien gespeichert sind, haben), können Sie den Text als normalen Kommentar ins Ticket schreiben:

```
workflowApi.addTicketText('This is my debug message', 'This is the subject of my
 debug message', false)
```

D.16.2.3 Debug der ConSol CM-Standardskripte

In ConSol CM-Standardskripten z. B. *createTicket.groovy* finden Sie Anweisungen wie die Folgenden:

```
if (log.isDebugEnabled()) {
   log.debug("Extracted email from from-field is $email")
}
```

Code-Beispiel 93: *Debug-Eintrag im Standard-E-Mail-Skript von ConSol CM*

Um die Debug-Ausgabe zu aktivieren, d. h. damit CM Debug-Informationen in die Log-Datei schreibt, müssen Sie das Log-Level des entsprechenden Moduls (hier: E-Mail) auf *DEBUG* setzen. Dies erfolgt in der Datei `jboss-log4j.xml`.

Wir gehen an dieser Stelle nicht näher auf dieses Thema ein. Weitere Informationen über CM-Logging finden Sie im *ConSol CM Operations Manual*.

E - Best Practices

In diesem Kapitel werden folgende Themen behandelt:

E.1 Die grundlegende Organisation eines Workflows: Verwendung von Bereichen

Eines der ersten Dinge, die Sie bei der Entwicklung des Konzepts für einen Workflow berücksichtigen müssen, ist die Anzahl und Anordnung der Bereiche (Scopes). In der Einführung in Bereiche können Sie Ihr Wissen über Bereiche auffrischen.

> Natürlich können Sie den Workflow zu einem späteren Zeitpunkt immer noch ändern, was sich aber auf bereits vorhandene Tickets, Sichten und Reports auswirken kann. Dies ist insbesondere dann wichtig, wenn der Workflow in einer Produktionsumgebung eingesetzt wird.

Berücksichtigen Sie folgende Punkte beim Aufsetzen der Grundstruktur eines Workflows:

- Welcher Trigger soll für das Ticket in welchen Zuständen des Prozesses aktiv sein?
  Soll zum Beispiel ein Zeit-Trigger, der neue Tickets überwacht, auch für Tickets aktiv sind, die schon bearbeitet werden? Oder soll ein Mail-Trigger aktiv sein, nachdem das Ticket vom Bearbeiter abgeschlossen wurde?
- Welche Sichten sind erforderlich?
  Sichten basieren auf der Position von Tickets in Bereichen. Details dazu finden Sie im *ConSol CM Administratorhandbuch* Abschnitt *Sichtenverwaltung*.

E.1.1 Variante A: Verwenden eines globalen Bereichs

Ein globaler Bereich ist ein Bereich, der alle anderen Bereiche des Workflows enthält. Möglicherweise möchten Sie so einen globalen Bereich verwenden, weil in einigen Prozessen während des gesamten Prozesses auf Events reagiert werden muss. Diese Events werden mithilfe von Triggern implementiert, die an den globalen Bereich angehängt werden. Wenn Sie zum Beispiel im gesamten Prozess überwachen möchten, ob eine E-Mail eingegangen ist, hängen Sie einen Mail-Trigger (siehe Abschnitt Mail-Trigger) an den globalen Bereich. Alle Unterbereiche des globalen Bereichs erben die Überwachung durch diesen Trigger. Wenn die E-Mails nur für einen Unterbereich überwacht werden sollen, können Sie den Mail-Trigger an diesen Unterbereich anhängen.

Das gleiche gilt für alle anderen Trigger, d. h. Event-Trigger (siehe Abschnitt Event-Trigger) und Zeit-Trigger (siehe Abschnitt Zeit-Trigger).

Der Startknoten sollte immer außerhalb des globalen Bereichs positioniert werden.

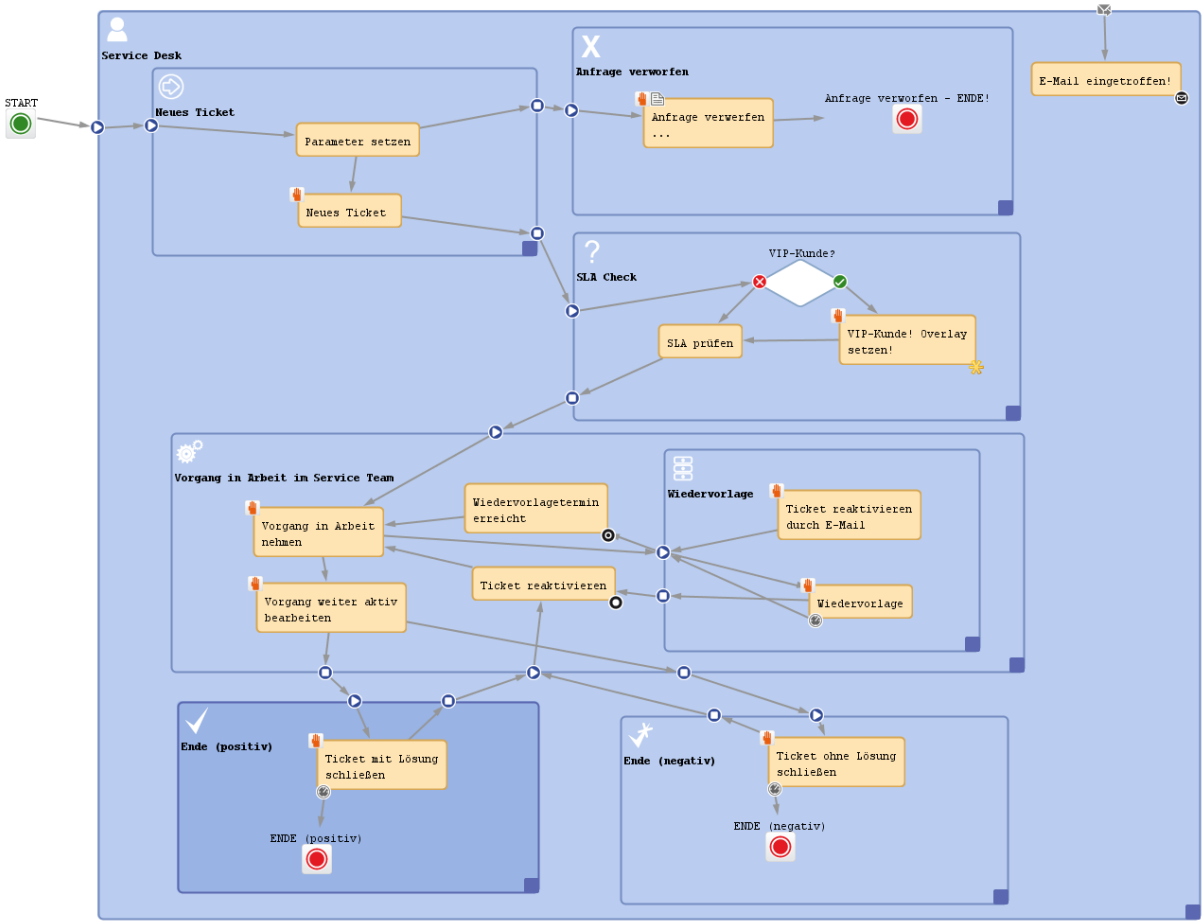


Abbildung 167: *ConSol CM Process Designer - Workflow mit globalem Bereich*

Denken Sie daran, dass Sie in inneren Bereichen immer Trigger verwenden können, die das Ereignis verbrauchen (siehe Abschnitt Reihenfolge des Feuerns von Event-Triggern in hierarchischen Bereichen als Beispiel für Event-Trigger). Wenn Sie zum Beispiel einen Mail-Trigger für den gesamten Prozess im

globalen Bereich verwenden möchten, aber im Bereich *Abgeschlossen* eine bestimmte Reaktion des Tickets benötigen, können Sie zusätzlich einen Mail-Trigger verwenden, der am Bereich *Abgeschlossen* hängt.

Variante B: Verwenden von drei oder mehr Hauptbereichen

Ein alternativer Weg zum Aufbauen eines Workflows ist die Verwendung von drei oder mehr Hauptbereichen:

- Neue Tickets
- In Bearbeitung (nur hier wird ein Mail-Trigger angewendet)
- Geschlossene Tickets (in einem oder mehreren eigenen Bereichen)

Die folgende Abbildung zeigt ein Beispiel für einen Workflow, der nach diesem Prinzip aufgebaut ist.

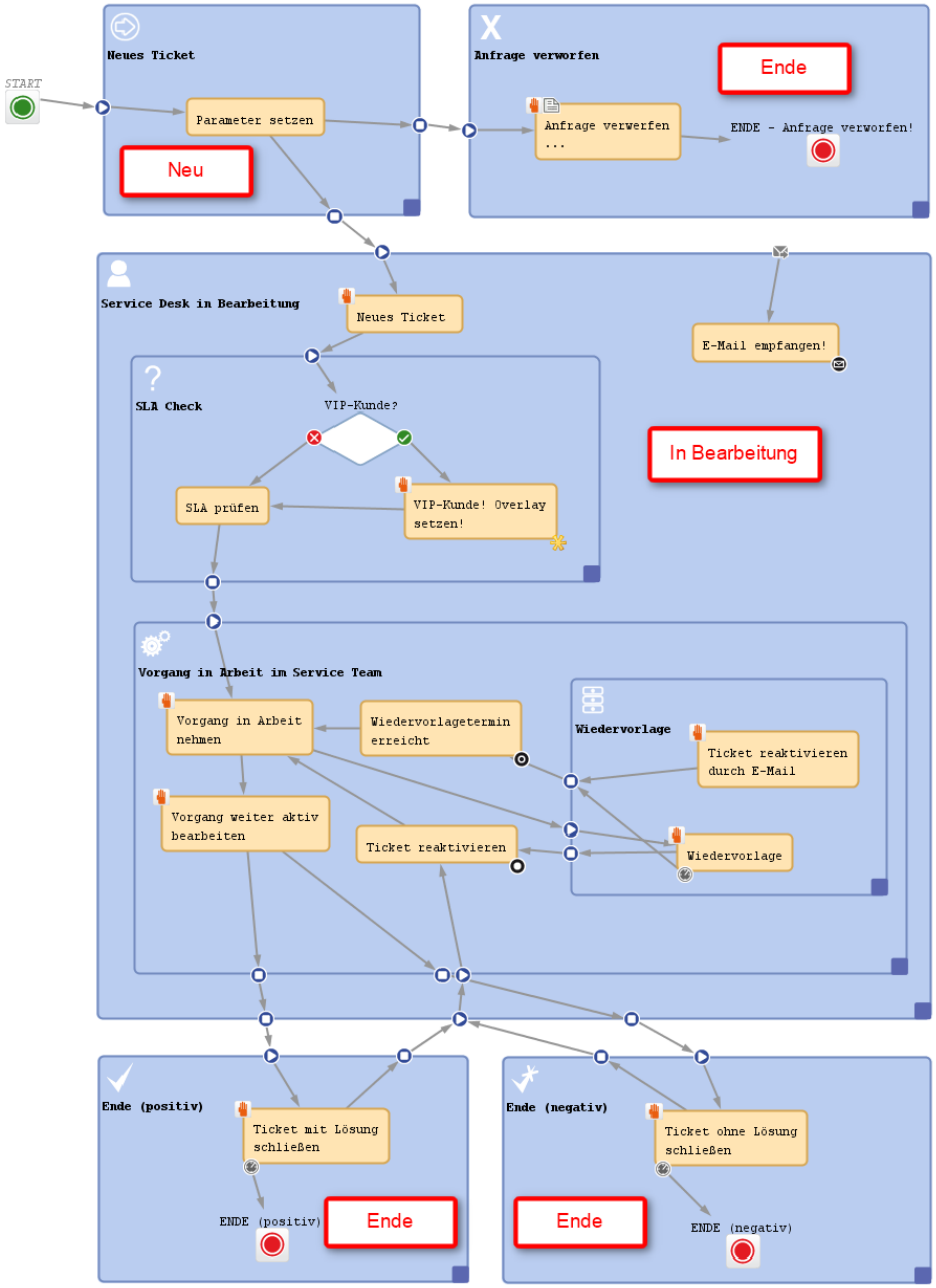


Abbildung 168: *ConSol CM Process Designer - Workflow mit drei Arten von Hauptbereichen*

E.2 Die Position des Startknotens

Die beste Postion für den Startknoten (siehe Abschnitt Workflow-Komponenten: Startknoten) hängt von der Verwendung von Triggern im auf den Knoten folgenden Bereich ab. Wenn im ersten Bereich, in den die Tickets nach dem Startknoten wandern, Zeit-Trigger verwendet werden, sollte der Startknoten außerhalb des Bereichs platziert werden. Wenn sich der Startknoten im ersten Bereich befindet, wird der Zeit-Trigger möglicherweise nicht richtig initialisiert. Platzieren Sie den Startknoten also im Standardbereich.

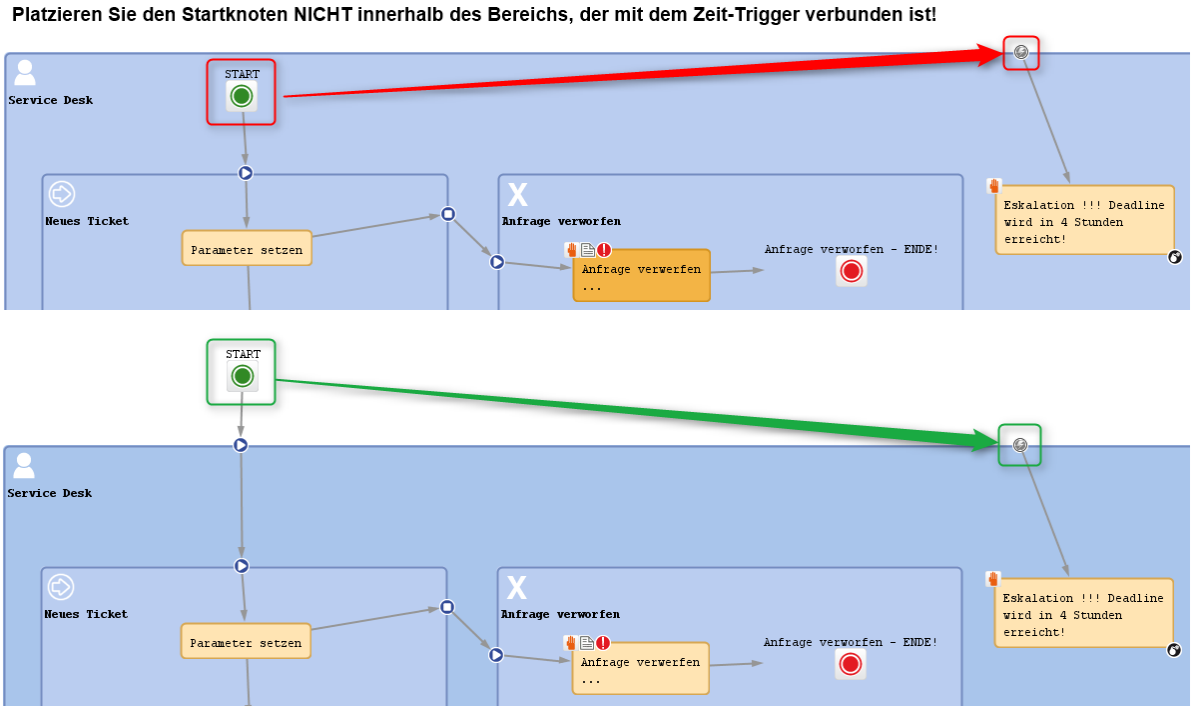


Abbildung 169: *ConSol CM Process Designer - Position des Startknotens*

E.3 Speichern einiger Workflow-Skripte im Admin Tool

Skripte, die immer wieder in Workflow-Aktivitäten und/oder Bedingungsskripten verwendet werden, sollten im Abschnitt *Skripte* des Admin Tools gespeichert und aus dem Workflow aufgerufen werden.

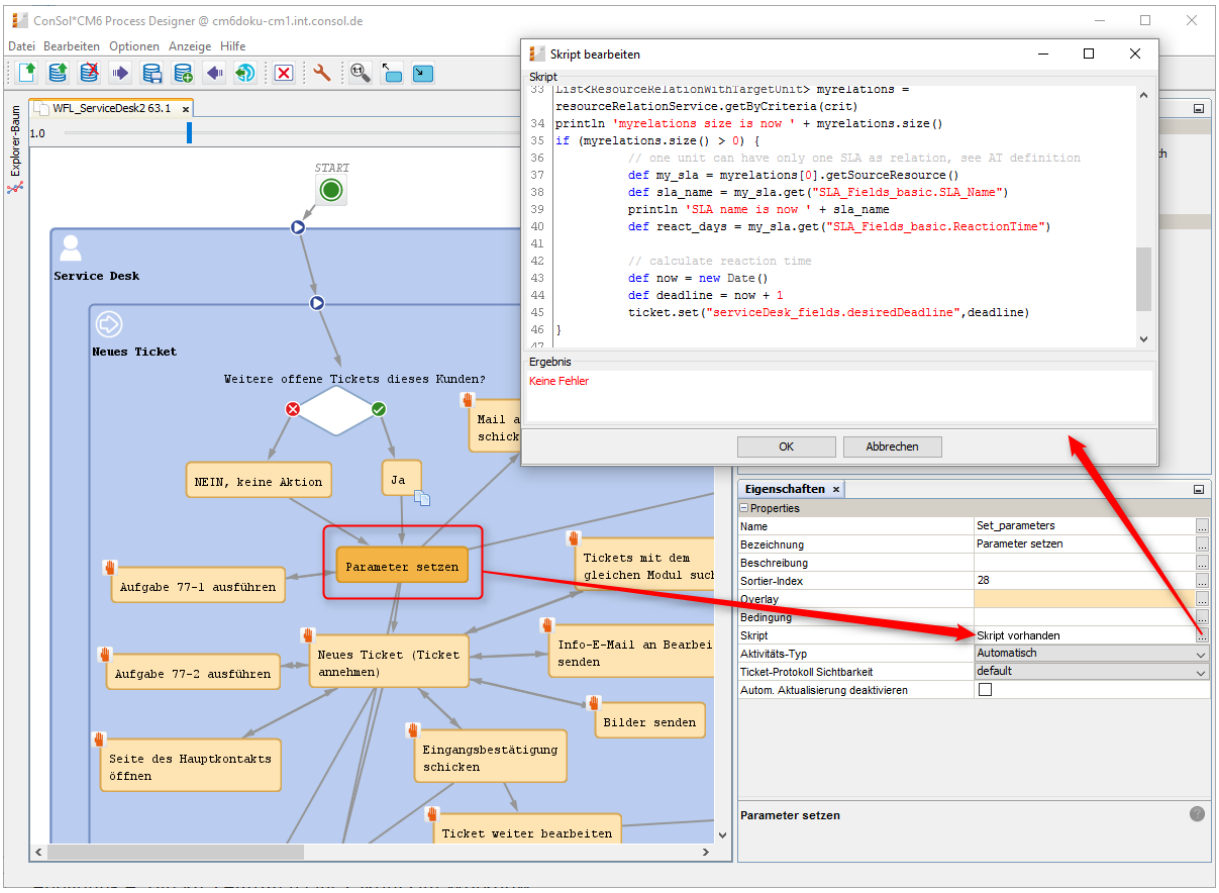


Abbildung 170: *Direktes Aufrufen eines Skripts im Workflow*

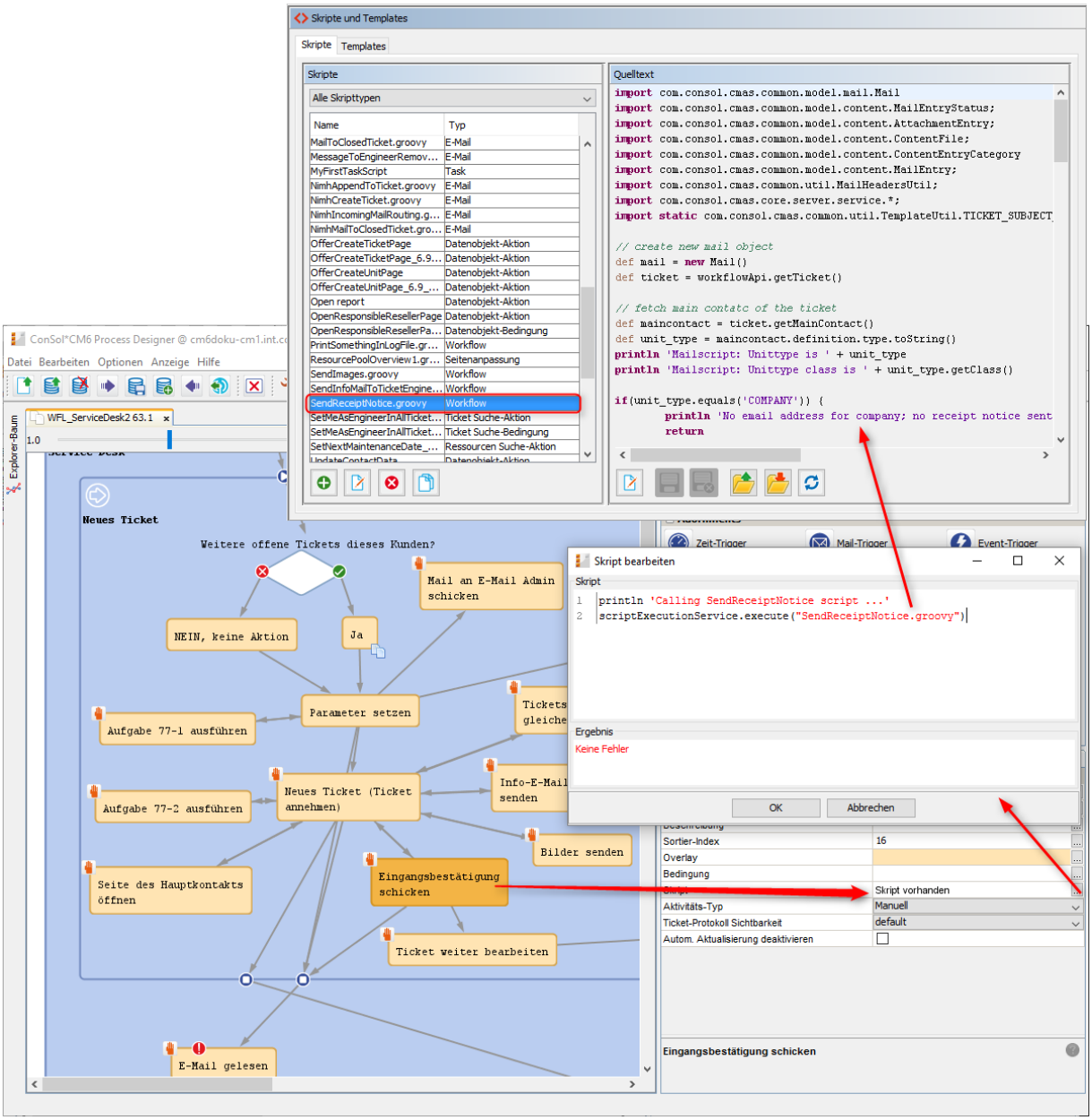


Abbildung 171: *Aufrufen eines Admin-Tool-Skripts aus einer Workflow-Aktivität*

E.3.1 Wann im Admin Tool gespeicherte Workflow-Skripte verwendet werden

Wir empfehlen weder diese Methode immer zu verwenden, noch raten wir von dieser Methode ab. Stattdessen zeigen wir Ihnen die Vor- und Nachteile dieses Ansatzes und Sie können selber entscheiden, wo Sie sie in Ihrem System anwenden möchten.

Die **Vorteile** des Speicherns von Workflow-Skripten im Admin Tool sind:

- Das Skript wird nur einmal gespeichert und muss nur an einer Stelle gepflegt/geändert werden.
- Änderungen am Skript werden im System sofort ausgeführt; es ist keine Installation (wie bei Workflows) erforderlich.

Die **Nachteile** des Speicherns von Workflow-Skripten im Admin Tool sind:

- Die Prozesslogik ist an zwei Stellen gespeichert, d. h. Sie müssen immer sowohl im Process Designer als auch im Admin Tool arbeiten, um den gesamten Prozess zu sehen.
- Der Skripteditor im Admin Tool ist nicht so bequem wie der Workflow-Skripteditor.
- Die meisten Objekte müssen in Admin-Tool-Skripte importiert werden, da sie nicht implizit vorhanden sind.
- Ein Workflow-Export alleine reicht nicht aus, um einen Workflow zu übertragen, da die Admin-Tool-Skripte nicht im Export enthalten sind.

E.3.2 Wie im Admin Tool gespeicherte Workflow-Skripte verwendet werden

Admin-Tool-Skripte, die im Workflow verwendet werden, müssen vom Typ *Workflow* sein. Ein Admin-Tool-Skript wird immer über die Interface *ScriptProvider* aus dem Workflow aufgerufen.

```
def scriptProvider = scriptProviderService.createDatabaseProvider
 ("scriptName.groovy")
def r = scriptExecutionService.execute(scriptProvider)
```

Code-Beispiel 94: *Aufrufen eines Admin-Tool-Skripts aus dem Workflow (einziger Weg in CM-Versionen 6.10.4 und älter, in CM 6.10.5 und höher noch verfügbar)*

```
// Create the scriptProvider for the required Admin Tool script, here
 "scriptName.groovy"
def scriptProvider = scriptProviderService.createDatabaseProvider
 ("scriptName.groovy")
// Define a HashMap with the key-value pairs which you would like to pass to the
 Admin Tool
def params = [ "templateName": "newCustomer" ]

// Execute the script. The passed parameters are available in the Admin Tool
 script. In the
// example, the variable templateName does not have to be defined in the Admin Tool
 script
// but it is present based on the definition in the passed HashMap.
// The variable r will contain the return value of the script or Null if there is
 no return
// value
def r = scriptExecutionService.execute(scriptProvider, params)
```

Code-Beispiel 95: *Aufrufen eines Admin-Tool-Skripts aus dem Workflow unter Verwendung von Parametern (einziger Weg in CM-Versionen 6.10.4 und älter, in CM 6.10.5 und höher noch verfügbar)*

```
scriptExecutionService.execute("MyScript")
```

Code-Beispiel 96: *Aufrufen eines Admin-Tool-Skripts aus dem Workflow (nur CM-Versionen 6.10.5 und höher)*

Da das Objekt workflowApi (siehe Abschnitt workflowAPI) in den Admin-Tool-Skripten nicht verfügbar ist, müssen Sie andere Klassen mit Methoden finden, die Sie anstelle der Methoden von workflowAPi verwenden können.

E.4 Optimieren von Trigger-Kombinationen

Vermeiden Sie das unnötige Ausführen von Triggern! Es verbraucht Ressourcen und verlangsamt die Performance der Applikation.

Beispiel 1:
Dieses Beispiel zeigt viele Event-Trigger in einem großen *globalen* Bereich.

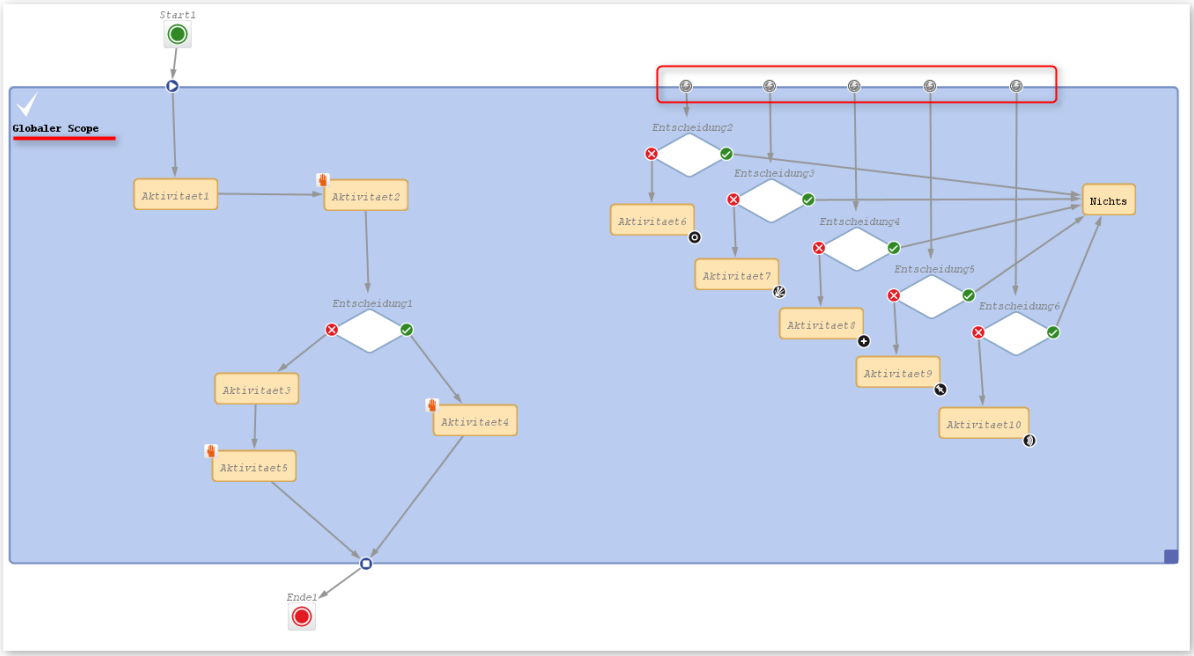


Abbildung 172: *ConSol CM Process Designer - Bereich mit Triggern*

Beispiel 2:
Verwenden Sie Trigger wenn möglich im kleinstmöglichen Bereich (in diesem Beispiel wurde der Trigger für *Entscheidung6* in einen kleineren Bereich verschoben).

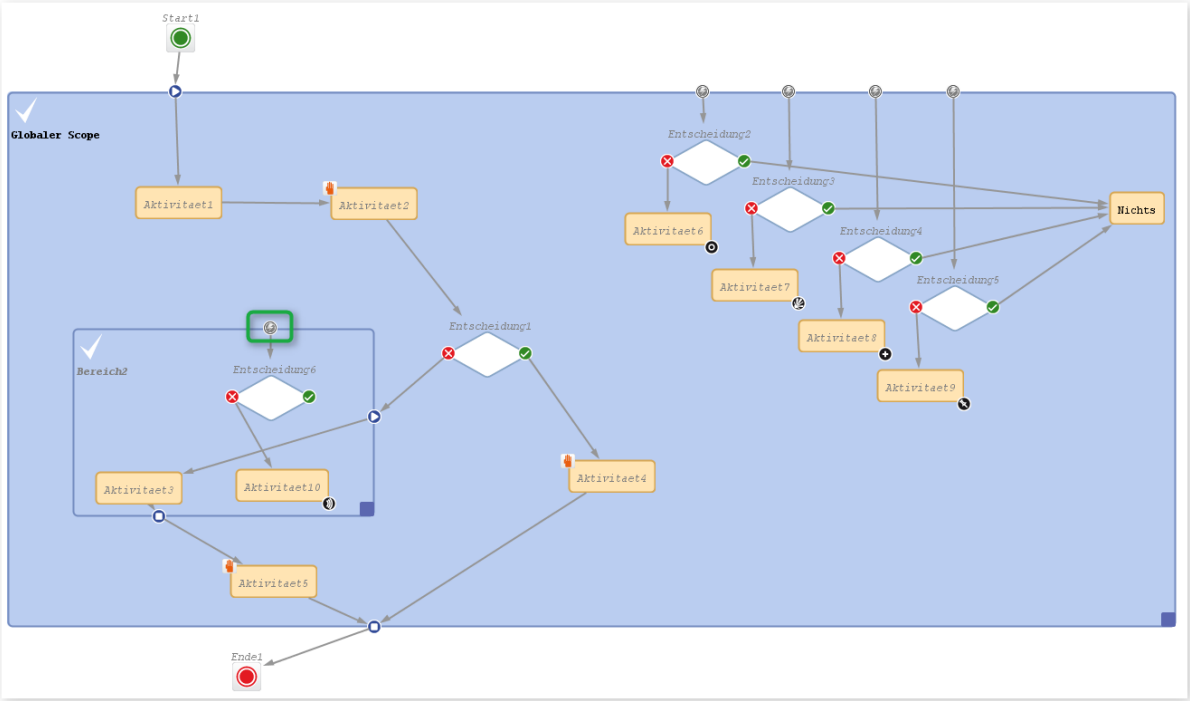


Abbildung 173: *ConSol CM Process Designer - Verschieben eines Triggers in einen kleineren Bereich*

Beispiel 3:
Wenn es **nicht** möglich ist, Trigger in kleinere Bereiche zu verschieben und Sie nicht alle Trigger aufrufen möchten, wenn eine Aktivität ausgeführt wird, verschieben Sie diese Aktivität in einen äußeren Bereich ohne Trigger.

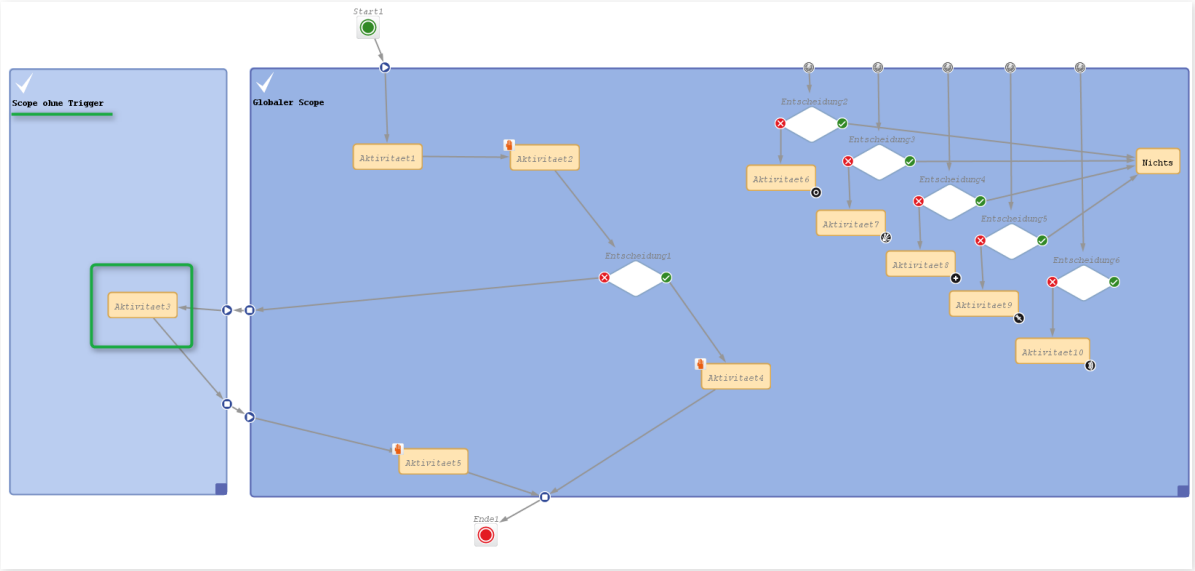


Abbildung 174: *ConSol CM Process Designer - Separate Bereiche mit und ohne Trigger*

In diesem Beispiel wurde die Position von *Aktivität3* optimiert. Sie hat vorher viele *Entscheidungen* angestoßen, die alle bei *Nichts* landeten. Die Ausführung von *Aktivität3* außerhalb des globalen Bereichs sorgt für eine gute Qualität der Workflow-Performance!

E.5 Anstoßen von Ticket-Update-Events nur wenn wirklich erforderlich

⚠ Vermeiden Sie unnötige Ticket-Update-Events (Java-Klasse *TicketUpdateEvent*)!

Das Zuweisen des aktuellen Bearbeiters (der Bearbeiter, der angemeldet ist und im Web Client arbeitet) zu einem Ticket kann auf zwei Wegen erfolgen. Bei einer Lösung wird ein Ticket-Update-Event angestoßen, bei der anderen Lösung nicht. Wenn es für den Anwendungsfall nicht erforderlich ist, einen *TicketUpdateEvent* anzustoßen, sollten Sie dies vermeiden, da unnötige Aufrufe von *TicketUpdateEvent* die Performance verschlechtern.

```
//this method throws a TicketUpdateEvent after assigning the current engineer to
 the ticket
workflowApi.assignEngineer(workflowApi.currentEngineer)
```

Code-Beispiel 97: *Code, der ein TicketUpdateEvent anstößt*

```
//this method does NOT throw a TicketUpdateEvent!
ticket.setEngineer(workflowApi.currentEngineer)
```

Code-Beispiel 98: *Code, der kein TicketUpdateEvent anstößt*

E.6 Verwenden des Parameters zum Deaktivieren von automatischen Aktualisierungen

⚠ Seien Sie vorsichtig bei der Verwendung des Flags *Autom. Aktualisierung deaktivieren*!

Denken Sie daran, dass standardmäßig nach der jeder Ausführung einer Aktivität ein Ticket-Update-Event angestoßen wird. Ein Ticket-Update-Event ist eine Operation mit großen Auswirkungen, die vorsichtig eingesetzt werden sollte!

Um Performance-Probleme zu vermeiden, können Sie den Flag *Autom. Aktualisierung deaktivieren* setzen. Es hängt von der Geschäftslogik ab, ob es Sinn macht, den Flag zu verwenden oder nicht.

In einer Kette mit automatischen Aktivitäten ist eine gute Vorgehensweise zum Beispiel:

- Bei der **1.** automatischen Aktivität ist der Flag *Autom. Aktualisierung deaktivieren* **eingeschaltet**.
  (Die Methode des Ticket-Update-Service wird nach der Ausführung der Aktivität **nicht** aufgerufen.)
- Bei der **2.** automatischen Aktivität ist der Flag *Autom. Aktualisierung deaktivieren* **eingeschaltet**.
  (Die Methode des Ticket-Update-Service wird nach der Ausführung der Aktivität **nicht** aufgerufen.)
- Bei der **3.** automatischen Aktivität ist der Flag *Autom. Aktualisierung deaktivieren* **eingeschaltet**.
  (Die Methode des Ticket-Update-Service wird nach der Ausführung der Aktivität **nicht** aufgerufen.)
  ...
- Bei der **letzten** automatischen Aktivität ist der Flag *Autom. Aktualisierung deaktivieren* **ausgeschaltet**.
  (So **wird** das *TicketUpdateEvent* **einmal** aufgerufen, am **Ende** der Kette!)

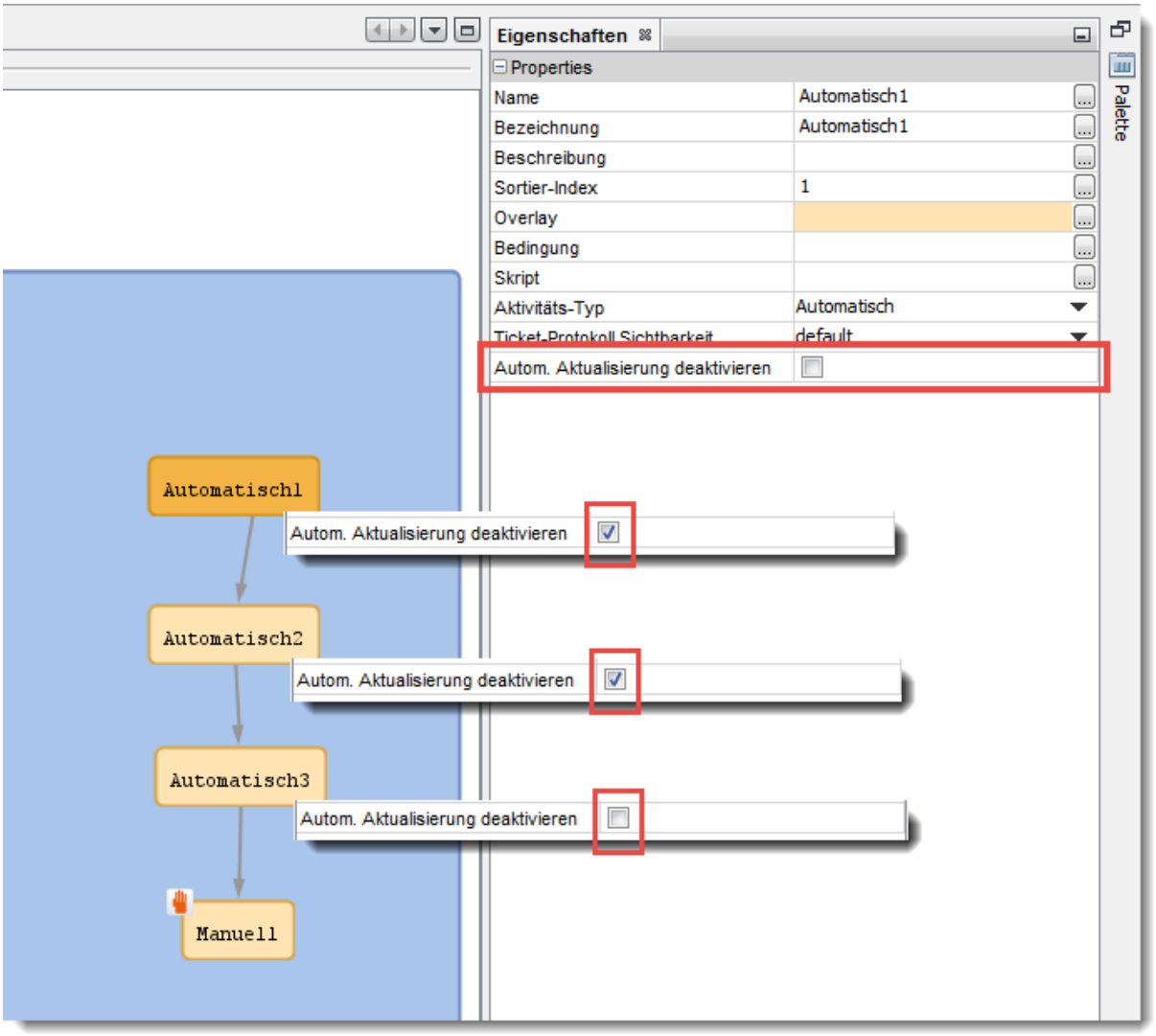


Abbildung 175: *ConSol CM Process Designer - Aktivitäten mit der Option "Autom. Aktualisierung deaktivieren"*

E.7 Vermeiden von sich selbst auslösenden Event-Triggern

Wenn Sie einen Event-Trigger verwenden, auf den eine automatische Aktivität folgt, achten Sie darauf, dass in dieser automatischen Aktivität die Felder oder Objekte, die den Event-Trigger auslösen, **nicht** noch einmal geändert werden (wodurch der Trigger erneut feuern würde)!

Wenn der Anwendungsfall vorsieht, dass die Felder, die das Feuern des Triggers ausgelöst haben, erneut geändert werden müssen, muss die Logik, mit der die Felder geändert werden, außerhalb des Bereichs, in dem sich der Trigger befindet, platziert werden.

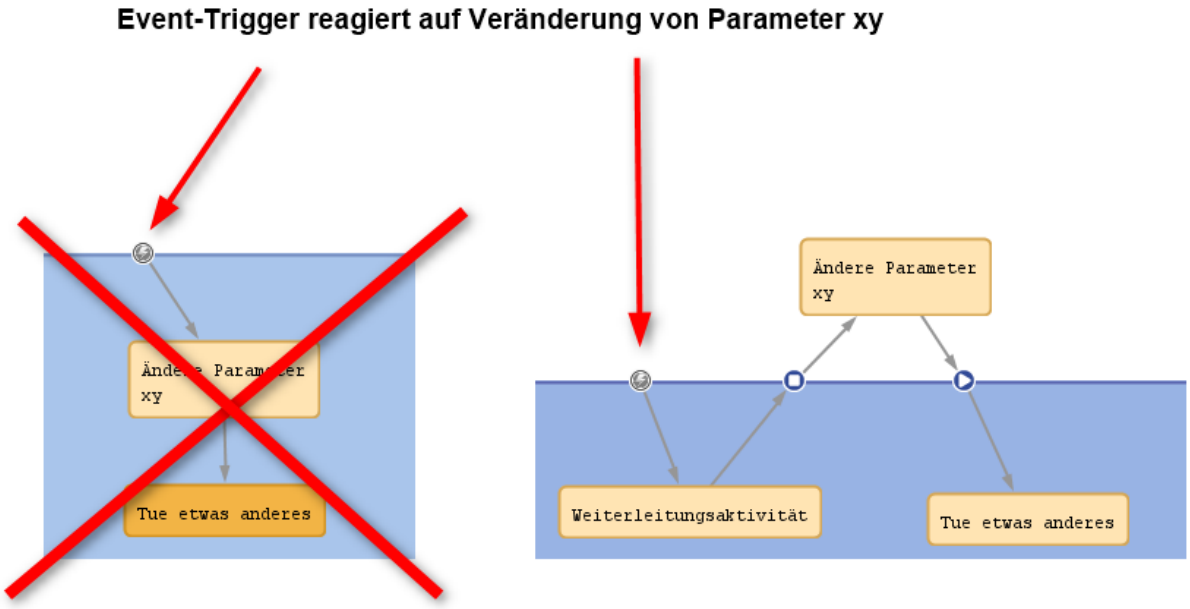


Abbildung 176: *ConSol CM Process Designer - Vermeiden von sich selbst auslösenden Event-Triggern*

F - Installieren von Workflows

In diesem Kapitel werden folgende Themen behandelt:

F.1 Einführung und Lebenszyklus eines Workflows

Während der Entwicklung eines Workflows verwenden Sie folgende Funktionen, die den Lebenszyklus eines Workflows widerspiegeln:

- Klicken Sie auf den Button *Laden ...*, um den Workflow zu laden oder erstellen Sie einen neuen Workflow, z. B. Version 1.2.
- Bearbeiten Sie den Workflow.
- Klicken Sie auf *Speichern als neue Version*, um den Workflow als neue Version zu speichern. Es wird eine neue Versionsnummer verwendet, z. B. 2.0.
- Fahren Sie mit der Bearbeitung des Workflows fort.
- Klicken Sie auf den Button *Speichern ...*, um den Workflow in der aktuellen Version zu speichern, z. B. Version 2.0.
- Fahren Sie mit der Bearbeitung des Workflows fort.
- Klicken Sie auf den Button *Installieren*, um den Workflow zu installieren. Damit wird der Workflow *gespeichert* und *installiert*, z. B. in Version 3.0.
  Ein installierter Workflow hat immer eine höhere Hauptversionsnummer als die letzte gespeicherte Version.
  Der Workflow, der vorher aktiv/installiert war, ist nicht mehr aktiv. Stattdessen ist sofort die neue Version des Workflows in Betrieb. Das ConSol CM-System muss nicht heruntergefahren werden.
  Die neue Version ist in der Liste der Workflows, die für die Vorgänge *Laden* und *Löschen* geöffnet wird, fett markiert und hat den Status *ist aktuell installiert*.
  Nach diesem Schritt wird die nächste Version als *neue Version* gespeichert.

Stellen Sie sicher, dass Sie wissen, wie viele Tickets übertragen werden müssen, wenn ein neuer Workflow installiert wird! Der Installationsvorgang kann in großen Umgebungen einige Zeit dauern! Siehe Abschnitt Aktionen während der Workflow-Installation.

F.2 Für die Workflow-Installation erforderliche Bearbeiterberechtigungen

Ein Bearbeiter, der Workflows installieren soll, muss mindestens eine Rolle mit einer der folgenden Zugangsberechtigungen haben:

- **Allgemeine Berechtigungen:**
  Administrator
- **Workflow-Berechtigungen:**
  Workflow installieren

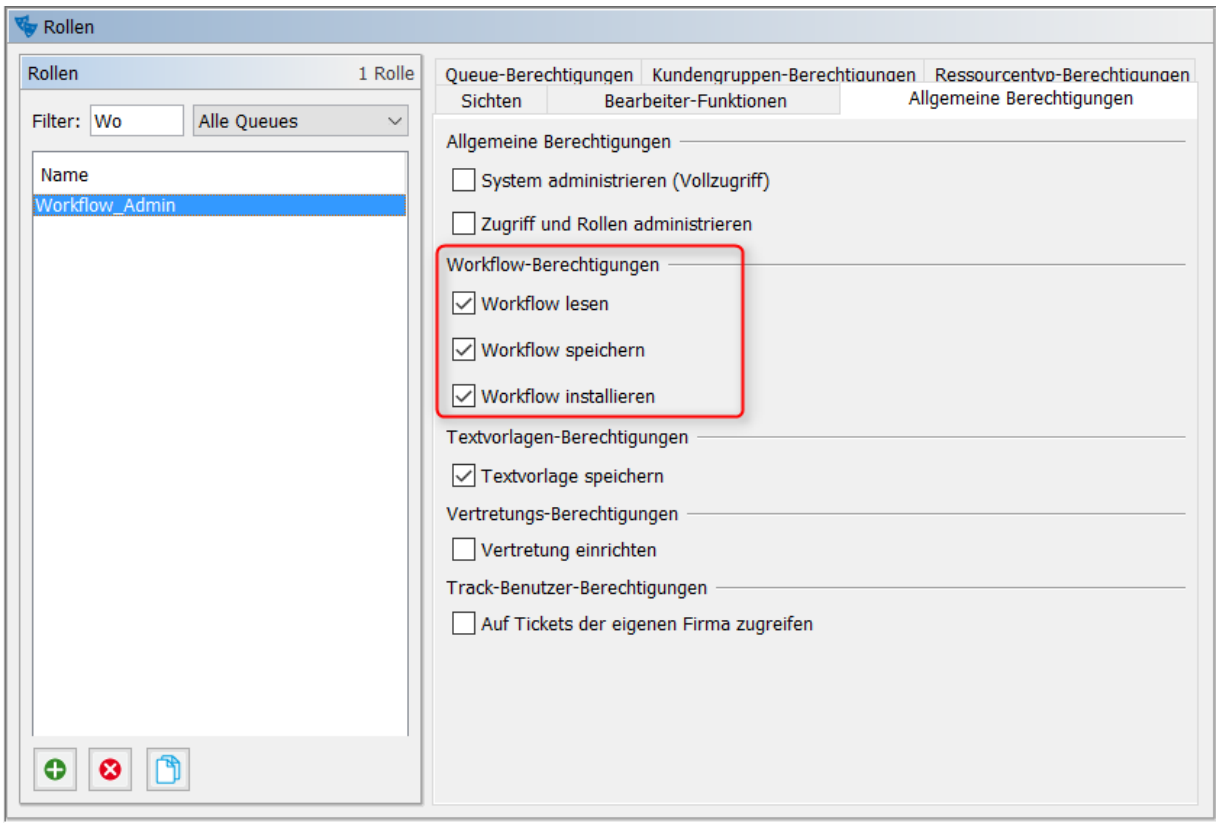


Abbildung 177: *ConSol CM Admin Tool - Bearbeiterberechtigungen für die Installation von Workflows*

F.3 Aktionen während der Workflow-Installation

Wenn ein Workflow installiert wird, ist er sofort aktiv. Bedenken Sie also, was mit den offenen Tickets geschehen soll, die sich in einer Queue befinden, auf die der neue Workflow angewendet wird. Diese Tickets werden in den neuen Workflow übertragen.

Falls Sie einen oder mehrere der folgenden Schritte durchgeführt haben:

- Entfernen einer oder mehrerer Aktivitäten
- Hinzufügen einer oder mehrerer automatischer Aktivitäten
- Hinzufügen eines oder mehrerer Trigger

werden die folgenden Aktionen initialisiert, nachdem Sie auf den Button *Installieren* geklickt haben.

Sie werden aufgefordert, zu entscheiden, was mit den offenen Tickets in den entsprechenden Queues geschehen soll, die aufgrund der Änderung der Workflow-Architektur ihre bisherige Position im Prozess nicht beibehalten können:

1. Die Tickets verbleiben so nah wie möglich an ihrer vorherigen Position (Standard).
2. Alle Tickets starten den Prozess von Beginn an.

Wenn Sie die erste Option (*Position im Prozess beibehalten*) auswählen, werden folgende Aktionen durchgeführt:

1. Die Übertragung der Tickets beginnt.
2. Der Name der letzten Aktivität, die für ein Ticket ausgeführt wurde, wird mit den Namen in der aktuellen Workflow-Definition verglichen. Wenn die Aktivität des Tickets nicht mehr in der Workflow-Definition enthalten ist, muss eine neue Zielaktivität für das Ticket gefunden werden.
3. Das *Ticketprotokoll* wird geladen. Die Übertragungs-Engine iteriert über alle seit Beginn der Prozessinstanz ausgeführten Aktivitäten und sucht nach einer passenden Aktivität, d. h. einer Aktivität, die
    a. noch in der Workflow-Definition enthalten ist,
    b. nicht das Zielelement eines Triggers ist,
    c. keine Sackgasse darstellt.

Alle Tickets, die ihre Position im Prozess nicht beibehalten können, werden nach diesen Kriterien an eine passende Position verschoben. In jedem Fall werden die Tickets im Workflow nach hinten und nie nach vorne verschoben.

Um eine *Zusammenfassung* aller Ticket-Übertragungen anzuzeigen, klicken Sie im Hauptmenü auf *Anzeige* und wählen Sie *Zeige die Übertragungshistorie der Tickets an*:

- **Workflow-Name**
  Name des Workflows.
- **Version**
  Version des alten Workflows.
- **Startzeit**
  Start der Übertagung. Ist die Startzeit des *Installationsvorgangs*.

- **Endezeit**
  Ende der Übertragung. Nach dieser Zeit ist der neue Workflow vollständig in Betrieb.
- **Übertragene Tickets**
  Anzahl der Tickets, die übertragen wurden, d. h. die während der Workflow-Installation vom System angefasst wurden. Sollte identisch sein mit der Summe der offenen Tickets aus allen Queues, die diesen Workflow benutzen.
- **Details**
  Zusätzliche Informationen bezüglich der Installation mit Ticketübertragung.

In der unteren rechten Ecke der GUI des Process Designers wird der Gesamtstatus der Ticketübertragung angezeigt.

G - Appendix

Dieser Abschnitt enthält mehrere Appendizes:

G.1 Annotationen

Es gibt zwei Arten von Annotationen: Feldannotationen und Gruppenannotationen. Feldannotationen werden auf ein einzelnes Ticket-, Kunden oder Ressourcenfeld angewendet. Gruppenannotationen werden auf eine Ticket-, Kunden oder Ressourcenfeldgruppe angewendet. Siehe:

- Liste der Feldannotationen
- Liste der Gruppenannotationen

G.1.1 Liste der Feldannotationen

G.1.1.1 cmweb-common (Typ)

groupable

- **Typ**: cmweb-common
- **Beschreibung**: Ermöglicht die Gruppierung in der Ticketliste.
- **Werte**:
  - *true*: Wird nur für Datenfelder des Typs *enum* verwendet. Entfernen Sie die Annotation, wenn Sie die Gruppierung deaktivieren möchten.

sortable

- **Typ**: cmweb-common
- **Beschreibung**Ermöglicht die Sortierung in der Ticketliste.
- **Werte**:
  - *true*: Wird für Datenfelder des Typs *date* oder *enum* verwendet. Entfernen Sie die Annotation, wenn Sie die Sortierung deaktivieren möchten.
    Für Felder des Typs *enum* : Funktioniert nur, wenn alle Werte des *enum*-Feldes indiziert werden.

G.1.1.2 common (Typ)

leave-trailing-zeros

- **Typ**: common
- **Beschreibung**: Wird für die Anzeige von Festkommazahlen verwendet.
- **Werte**:
  - *true / false*: Nullen am Ende der Nachkommastellen werden nicht abgeschnitten, wenn der Wert *true* ist.

readonly

- **Typ**: common
- **Beschreibung**Zeigt an, dass das Benutzerdefinierte Feld nicht verändert werden kann.
- **Werte**:
  - *true / false*: Das Feld ist schreibgeschützt, wenn der Wert auf *true* gesetzt ist. Wenn kein Wert oder ein anderer Wert als *false* gesetzt ist, wird *true* angenommen.

visibility

- **Typ**: common
- **Beschreibung**: Definiert, wann das Feld sichtbar ist.

- **Werte**:
  - *edit*: Das Feld wird im Bearbeitungsmodus angezeigt.
  - *view*: Das Feld wird im Ansichtsmodus angezeigt.
  - *none*: Das Feld ist nicht sichtbar.
  - Wenn ein anderer Wert oder kein Wert gesetzt ist, ist das Feld immer sichtbar.

G.1.1.3 component-type (Typ)

boolean-type

- **Typ**: component-type
- **Beschreibung**: Definition des Layouts eines Boolean-Felds.
- **Werte**:
  - *checkbox* (Standardwert): Das Feld kann markiert werden (standardmäßig auf *false* gesetzt).
  - *radio*: 2 Radio-Buttons (ja/nein) zur Auswahl (es kann nur einer aktiv sein).
  - *select*: Drop-down-Feld mit 2 Werten (ja/nein).

enum-in-search-type

- **Typ**: component-type
- **Beschreibung**: Definiert, ob für ein Feld des Datentyps *enum* bei einer Suche über mehrere Werte gesucht werden kann.
- **Werte**:
  - *single* (Standardwert) / *multiple*: Wenn der Wert *multiple* gesetzt ist, kann über mehrere Werte gesucht werden.

enum-type

- **Typ**: component-type
- **Beschreibung**: Definition des Layouts der Listenansicht
- **Werte**:
  - *select* (Standardwert): Drop-down-Liste für die Auswahl.
  - *radio*: Liste mit Radio-Buttons für die Auswahl (es kann nur eine Option aktiv sein).
  - *autocomplete*: Drop-down-Liste für die Auswahl. Das Feld ist ein Eingabefeld, mit dem die Liste gefiltert werden kann.

list-type

- **Typ**: component-type
- **Beschreibung**: Deaktiviert die Optionen zum Hinzufügen bzw. Löschen bei Datenfeldern des Typs *list* oder *struct*.

- **Werte**:
  - *fixed-size*: Es ist nicht möglich, Felder/Zeilen zu löschen oder hinzuzufügen.
  - *non-shrinkable*: Es ist nicht möglich, Felder/Zeilen zu löschen.
  - *non-growable*: Es ist nicht möglich, Felder/Zeilen hinzuzufügen.

text-type

- **Typ**: component-type
- **Beschreibung**: Definiert mögliche Arten eines Felds des Datentyps *string* .
- **Werte**:
  - *text* (Standardwert): Einzeiliges Eingabefeld
  - *textarea*: Mehrzeiliges Eingabefeld
  - *password*: Eingabefeld für Passwörter.
    Das Passwort wird im Ansichtsmodus als ******* angezeigt.
  - *label*: Eingabe wird als Label angezeigt, d.h. das Feld wird nur angezeigt, es ist keine Dateneingabe möglich.
  - *url*: Eingabe wird im Anzeige-Modus als Link dargestellt. Der String muss dafür diesem URL-Muster entsprechen:
    "^((?:mailto\:|(?:(?:ht|f)tps?)\://)1\S+)(?: (?:\|)?(.*))?$"
    Beispiel: "http://consol.de ConSol"
  - *file-url*: Die Eingabe wird als Link auf eine Datei im Dateisystem angezeigt. Der Webbrowser muss solche Links zulassen/unterstützen! Informationen dazu finden Sie im Abschnitt Details über String-Felder: Verwenden von Annotationen zum Anpassen von Strings. Dieser Link wird auch als Tooltip angezeigt.
    Die URL ist korrekt formatiert, wenn folgende Bedingungen erfüllt sind:
    Sie beginnt mit *file:* gefolgt von normalen Schrägstrichen:
    - drei Schrägstriche "///" für Dateien, die sich auf demselben Computer befinden wie der Browser (alternativ "//localhost/") oder
    - zwei Schrägstriche gefolgt vom Servernamen und einem weiteren Schrägstrich für Dateien auf Dateiservern, die vom Rechner, auf dem der Webbrowser läuft, erreichbar sind.

    Danach folgt der vollständige Dateipfad, der mit dem Dateinamen endet. Auf Microsoft Windows-Systemen wird der Pfad ebenfalls mit normalen Schrägstrichen anstelle von umgekehrten Schrägstrichen geschrieben.
    Der Laufwerksbuchstabe eines lokalen Pfads auf Microsoft Windows-Systemen wird wie üblich verwendet, zum Beispiel C:. Pfade mit Leerzeichen und Sonderzeichen wie "{, }, ^, #, ?" müssen auf Microsoft Windows-Systemen mit Prozentzeichen kodiert werden (z. B. mit "%20" für ein Leerzeichen).
    Beispiel-URLs:

- file://file-server/path/to/my/file.ext
- file:///linux/local/file.pdf
- file:///C:/Users/myuser/localfile.doc

G.1.1.4 contact authentication (Typ)

ldapid

- **Typ**: contact authentication
- **Beschreibung**: Wird in einer Datenobjektgruppe des Typs *Kontakt* für das Datenobjektgruppenfeld verwendet, das die LDAP-ID für die Authentifizierung in CM.Track enthält.
- **Werte**: Gibt an, dass das Feld im Authentifizierungsprozess als LDAP-ID verwendet wird. Als Datentyp ist *string* erforderlich.
  Da die Definition auf Kundengruppenebene vorgenommen wird, kann die LDAP-Authentifizierung im gemischten Modus verwendet werden, d. h. es kann für einige Kundengruppen LDAP verwendet werden und für andere Kundengruppen die reguläre Authentifizierung.

password

- **Typ**: contact authentication
- **Beschreibung**: Gibt an, dass das Feld im Authentifizierungsprozess als Passwort verwendet wird.
- **Werte**:
  - *<string>*: Für CM.Track verwendet.

username

- **Typ**: contact authentification
- **Beschreibung**: Gibt an, dass dieses Feld im Authentifizierungsprozess als Anmeldename verwendet wird.
- **Werte**:
  - *true / false*: Für CM.Track verwendet.

G.1.1.5 dwh (Typ)

dwh-no-history-field

- **Typ**: dwh
- **Beschreibung**: Die Annotation wird verwendet, um festzulegen, dass das Feld nicht im DWH protokolliert wird.
- **Werte**:
  - *true / false*: Seit Version 6.10.2.0.

reportable

- **Typ**: dwh
- **Beschreibung**: Gibt an, dass das Feld für Reports verwendet werden kann und an das DWH übertragen werden soll.
- **Werte**:
  - *true / false*: Das Feld kann in Reports verwendet werden, wenn der Wert auf *true* gesetzt ist.

G.1.1.6 indexing (Typ)

field indexed

- **Typ**: indexing
- **Beschreibung**: Gibt an, dass für das Feld ein Datenbankindex erstellt wird. Wenn es möglich sein soll, Ergebnistabellen (im Web Client) durch Klicken auf die Spaltenüberschrift nach einer Spalte zu sortieren, muss das entsprechende Feld indiziert sein!
- **Werte**:
  - *transitive* (Standardwert): Alle Daten werden angezeigt (Ticketdaten, Kundendaten und Ressourcendaten).
  - *unit*: Für Kundendaten verwendet. Es werden nur die Unit und die Parent Unit (d. h. Firma) als Suchergebnis zurückgegeben, Tickets werden nicht angezeigt.
  - *local*: Für Kundendaten verwendet. Es wird nur die Unit als Suchergebnis zurückgegeben, Firmen und Tickets werden nicht angezeigt.
  - *not indexed*: Das Feld ist nicht indiziert.

phonetic

- **Typ**: indexing
- **Beschreibung**: Aktiviert die phonetische Suche für dieses Feld. Kann nur für Datenfelder des Typs String verwendet werden (für long oder short string ebenfalls).
- **Werte**: *true*/*false*. Wird automatisch auf *true* gesetzt, wenn die Annotation hinzugefügt wird.

G.1.1.7 layout (Typ)

colspan

- **Typ**: layout
- **Beschreibung**: Legt fest, wie viele Spalten für das Feld im Layout reserviert werden.
- **Werte**:
  - *<Zahl>*: Anzahl der Spalten.

field-group

- **Typ**: layout
- **Beschreibung**: Ermöglicht die Gruppierung der Felder im Ansichtsmodus. Die Annotation wird im Bearbeitungsmodus ignoriert.
- **Werte**:
  - *<string>*: Um Felder zu gruppieren, muss für jedes Feld dieselbe Zeichenfolge als Annotation gesetzt werden. Zwei oder mehr Datenfelder sind verbunden, wenn sie für diese Annotation den gleichen Wert besitzen. Die Gruppe der verbundenen Datenfelder wird nur angezeigt, wenn in allen von ihnen Werte gesetzt sind.

fieldsize

- **Typ**: layout
- **Beschreibung**: Die angezeigte Feldgröße im Ticketlayout.
- **Werte**:
  - *<Zeilen>;<Spalten>*: Angezeigte Feldgröße.
    Format für Felder des Datentyps *string* und *number*: n gibt die Anzahl der Zeichen an. Für String-Felder ist das die Anzahl an Großbuchstaben M in einer Monospace-Schriftart).
    Format für *textarea*: Zeilen;Spalten (entspricht <textarea Zeilen="" Spalten="">).
    Sortierte Listen werden anstatt als Drop-down-Liste als Auswahlfeld mit n Elementen angezeigt. Format für Sortierte Listen: n.
    Hinweis: Diese Annotation dient nur zur Konfiguration des Layouts. Verwenden Sie für die Validierung *maxlength* vom Typ *group validation*.
  - *<Zahl>*: Für Datenfelder des Typs *enum*. Definiert, wie viele Werte im Listenfeld direkt sichtbar sind. Nur zu Layout-Zwecken.

label-group

- **Typ**: layout
- **Beschreibung**: Zeigt eine Gruppe von Feldern im Ansichtsmodus zusammen mit ihrem beschreibenden Label an. Die Annotation wird im Bearbeitungsmodus ignoriert.
- **Werte**:
  - *<string>*: Gibt eine Gruppe Datenfelder zusammen mit ihrem beschreibenden Label an. Die Annotation wird im Ansichtsmodus verwendet und im Bearbeitungsmodus ignoriert. Die Gruppe kann genau ein Label haben (ein Datenfeld des Typs *string* , dem die zusätzliche Annotation *text-type* mit dem Wert *label* zugewiesen ist). Das Label wird angezeigt, wenn für mindestens ein Datenfeld der Gruppe ein Wert gesetzt ist. Alle Felder mit demselben Label werden gruppiert und unter diesem Label angezeigt.
    Außerdem muss dem Label die Annotation *label-group* zugewiesen werden.

label-in-view

- **Typ**: layout
- **Beschreibung**: Zeigt den Wert des Datenfelds im Ansichtsmodus als Label an. Die Annotation wird im Bearbeitungsmodus ignoriert.
- **Werte**:
  - *true*: Entfernen Sie die Annotation, wenn das Label im Ansichtsmodus nicht angezeigt werden soll.

order-in-result

- **Typ**: layout
- **Beschreibung**: Zeigt das Feld in der Liste der Suchergebnisse als Spalte an der angegebenen Position an.
- **Werte**:
  - *<Zahl>*: Die Spalten sind aufsteigend sortiert.
    Seit CM-Version 6.0.1. Eine detaillierte Erklärung dazu finden Sie in der Infobox im Abschnitt *Konfiguration von Suche und Indexer* des *ConSol CM Administratorhandbuchs*.

position

- **Typ**: layout
- **Beschreibung**: Definiert die Position eines Feldes im Rasterlayout oder in einer Liste (*struct*).
- **Werte**:
  - *<Zahl>;<Zahl>*: Die Werte geben die *Zeile* und *Spalte* an (Zeile;Spalte), die Nummerierung beginnt bei 0;0. Wenn kein Wert gesetzt ist, wird das Datenfeld in der ersten freien Zelle des Rasters angezeigt.
  - *0;<Zahl>*: Es wird nur der Wert *Spalte* verwendet, der Wert *Zeile* wird ignoriert.

rowspan

- **Typ**: layout
- **Beschreibung**: Legt fest, wie viele Zeilen das Feld im Layout belegt.
- **Werte**:
  - *<Zahl>*: Anzahl der Zeilen.

show-label-in-edit

- **Typ**: layout
- **Beschreibung**: Ob das Datenfeld im Bearbeitungsmodus mit Label angezeigt werden soll.
- **Werte**:
  - *true / false*: Seit Version 6.9.4.

show-label-in-view

- **Typ**: layout
- **Beschreibung**: Ob das Datenfeld im Ansichtsmodus mit Label angezeigt werden soll.
- **Werte**:
  - *true / false*: Seit Version 6.9.4.

show-tooltip

- **Typ**: layout
- **Beschreibung**: Ob das Datenfeld als Tooltip angezeigt werden soll.
- **Werte**:
  - *true / false*: Seit Version 6.9.4.

show-watermark

- **Typ**: layout
- **Beschreibung**: Ob das Datenfeld mit Wasserzeichen angezeigt werden soll.
- **Werte**:
  - *true / false*: Seit Version 6.9.4.

ticket-list-colspan

- **Typ**: layout
- **Beschreibung**: Definiert wie viele Spalten das Feld im Ticketlistenfeld belegt.
- **Werte**:
  - *<Zahl>*: Anzahl der Spalten.

ticket-list-position

- **Typ**: layout
- **Beschreibung**: Definiert die Position des Feldes im Ticketlistenfeld.
- **Werte**:
  - *<Zahl>;<Zahl>*: Die Werte geben die *Zeile* und *Spalte* an (Zeile;Spalte), die Nummerierung beginnt bei 0;0.

ticket-list-rowspan

- **Typ**: layout
- **Beschreibung**: Definiert wie viele Zeilen das Feld im Ticketlistenfeld belegt.
- **Werte**:
  - *<Zahl>*: Anzahl der Zeilen.

G.1.1.8 performance (Typ)

no-history-field

- **Typ**: performance
- **Beschreibung**: Gibt an, dass ein einzelnes Datenfeld nicht protokolliert werden soll. Überschreibt die Gruppenannotation *no-history*.
- **Werte**:
  - *true / false*: Die Annotation ist aktiv, wenn der Wert auf *true* gesetzt ist. Für Felder, die gespeichert aber nicht im Ticketprotokoll angezeigt werden sollen, wird die Annotation *visibility configuration* verwendet.
    In CM-Versionen bis 6.10.2 wird auch die Übertragung der Feldhistorie an das DWH mit dieser Annotation gesteuert.
    Verwenden Sie dazu ab CM-Version 6.10.2 die Annotation *dwh-no-history-field*.

G.1.1.9 phone commander (Typ)

dialable

- **Type**: phone commander (CM.Phone)
- **Beschreibung**: Definiert ein Feld als Telefonnummer.
- **Werte**:
  - *true*: Für für CM.Phone verwendet. Markiert eine Telefonnummer als automatisch wählbar für ausgehende Anrufe im CTI-System.

G.1.1.10 resource (Typ)

resource-color

- **Typ**: resource
- **Beschreibung**:
- **Werte**:
  - *true / false*: Kann einem Feld des Datentyps *enum* mit einer Farbe zugewiesen werden. Die Farbe des ausgewählten Listenwerts wird als Hintergrundfarbe für das Ressourcen-Icon verwendet.

G.1.1.11 search result (Typ)

contact search result column

- **Typ**: search result
- **Beschreibung**: Gibt an, ob das Feld standardmäßig in den Suchergebnissen angezeigt werden soll. **Veraltet! Nicht mehr verwenden!**

- **Werte**:
  - *true*:
    Entfernen Sie die Annotation, wenn das Feld nicht standardmäßig angezeigt werden soll.

    Seit CM-Version 6.1.3
    (Ersetzt durch *order-in-result*, *contact search result column* ist veraltet!)

G.1.1.12 ticket contact relation type (Typ)

contains contacts

- **Typ**: ticket contact relation type
- **Beschreibung**: Wird nur für die Definition von Listenfeldern verwendet. Gibt an, dass das Feld Unit-Referenzen zu Kontakten enthalten kann.
- **Werte**:
  - *true/false*: Der Wert ist boolean. Gibt an, ob die Liste innerhalb eines Kontakts (true) oder eines Tickets (false) angezeigt wird.

G.1.1.13 ticket display (Typ)

enum field with ticket color

- **Typ**: ticket display
- **Beschreibung**: Legt die Hintergrundfarbe des Ticket-Icons in der Ticketliste und im Ticket fest.
- **Werte**:
  - *true / false*: Das Feld muss in den Sortierten Listen vorhanden sein, wo Listen, Werte und Farben definiert werden.

G.1.1.14 validation (Typ)

accuracy

- **Typ**: validation
- **Beschreibung**: Für Datumsfelder, um den Detailgrad der angezeigten Daten zu definieren
- **Werte**:
  - *date (Standardwert)*: Datum wird ohne Zeit angezeigt.
  - *date-time*: Datum wird mit Zeit angezeigt.
  - *only-time*: Es wird nur die Zeit nicht aber das Datum angezeigt.

email

- **Typ**: validation
- **Beschreibung**: Verwendet, um bei E-Mail-Adressen zu prüfen, ob das Format korrekt ist, d. h. ob es <name>@<domain> entspricht.

- **Werte**:
  - *true*: Kann für Datenfelder des Typs *string* verwendet werden. Entfernen Sie die Annotation, wenn das Format nicht validiert werden soll.

format

- **Typ**: validation
- **Beschreibung**: Verwendet, um das Format von Datumsfeldern zu validieren.
- **Werte**:
  - *<Datumsformat>*: Das Muster für das Datum basiert auf *SimpleDateFormat*, z. B. dd.MM.yyyy.
    Denken Sie daran, den richtigen Wert für *colspan* einzusetzen, wenn das Format Stunden/Minuten beinhaltet. Eine Formatreferenz finden Sie in http://docs.oracle.com/javase/6/docs/api/java/text/SimpleDateFormat.html.

matches

- **Typ**: validation
- **Beschreibung**: Prüft, ob die Eingabe in Datenfelder des Typs *string* einer angegebenen RegEx entspricht.
- **Werte**:
  - *<string>*: Kann bei Datenfeldern des Typs *string* verwendet werden.

maxLength

- **Typ**: validation
- **Beschreibung**: Definiert die maximale Länge der Eingabe in Datenfelder des Typs *string*.
- **Werte**:
  - *<Zahl>*: Kann für Datenfelder des Typs *string* verwendet werden.

maxValue

- **Typ**: validation
- **Beschreibung**: Definiert den Maximalwert für Datenfelder mit Zahlen.
- **Werte**:
  - *<Zahl>*: Kann für Datenfelder des Typs *number* verwendet werden, d. h. *number* und *fixed-point number*.

minLength

- **Typ**: validation
- **Beschreibung**: Definiert die minimale Länge der Eingabe in Datenfelder des Typs *string*.
- **Werte**:
  - *<Zahl>*: Kann für Datenfelder des Typs *string* verwendet werden.

minValue

- **Typ**: validation
- **Beschreibung**: Definiert den Minimalwert für Datenfelder mit Zahlen.
- **Werte**:
  - *<Zahl>*: Kann für Datenfelder des Typs *number* verwendet werden, d. h. *number* und *fixed-point number*.

required

- **Typ**: validation
- **Beschreibung**: Gibt an, dass es sich um ein Pflichtfeld handelt.
- **Werte**:
  - *true / false*: Das Feld ist ein Pflichtfeld, wenn der Wert auf *true* gesetzt ist. Der Benutzer kann das Ticket nicht speichern, ohne einen Wert in das Pflichtfeld eingegeben zu haben. Im Web Client sind Pflichtfelder mit einem roten Sternchen gekennzeichnet.

G.1.1.15 visibility (Typ)

visibility configuration

- **Typ**: visibility
- **Beschreibung**: Legt die Sichtbarkeit des Felds im Protokoll fest.
- **Werte**:
  - *on every level*: Das Feld wird im Protokoll in jedem Sichtbarkeitslevel angezeigt.
  - *2nd level and 3rd level*: Das Feld wird im 2. und 3. Sichtbarkeitslevel des Protokolls angezeigt.
  - *only 3rd level*: Das Feld wird nur im 3. Sichtbarkeitslevel des Protokolls angezeigt.

G.1.2 Liste der Gruppenannotationen

G.1.2.1 common (Typ)

group-visibility

- **Typ**common
- **Beschreibung**: Definiert die Standardsichtbarkeit einer Datenfeldgruppe.
- **Werte**:
    - *true / false*: Die Annotation kann auf Feldebene überschrieben werden.

G.1.2.2 dwh (Typ)

dwh-no-history

- **Typ**: dwh
- **Beschreibung**: Gibt an, dass alle Felder dieser Gruppe nicht im DWH protokolliert werden.
- **Werte**:
    - *true / false*: Seit Version 6.10.2.0

reportable group

- **Typ**: dwh
- **Beschreibung**: Gibt an, dass alle Datenfelder, die zu dieser Gruppe gehören, in Reports verwendet werden können und an das CMRF übertragen werden sollen.
- **Werte**:
    - *true / false*: Es muss ein Wert gesetzt werden. Die Annotation ist aktiv, wenn der Wert auf *true* gesetzt wird.

G.1.2.3 layout (Typ)

auto-open-group

- **Typ**: layout
- **Beschreibung**: Die Gruppe ist initial geöffnet. Es kann mehr als ein Wert als durch Kommas oder Semikolons getrennte Liste angegeben werden (kann für die Kundenannotation verwendet werden).
- **Werte**:
    - *ticket:create*: Die Gruppe ist initial geöffnet, wenn ein neues Ticket erstellt wird.
    - *customer:create*: Die Gruppe ist initial geöffnet, wenn ein neuer Kunde erstellt wird.
    - *customer:view*: Die Gruppe ist initial geöffnet, wenn die Kundenseite (Kontakt- oder Firmenseite) geöffnet wird.

show-contact-in-ticket-list

- **Typ**: layout
- **Beschreibung**: Veraltet! Verwenden Sie die Seitenanpassung! accordionTicketList.mainCustomerDescriptionVisible={true, false}
- **Werte**: veraltet

show-in-group-section

- **Typ**: layout
- **Beschreibung**: Legt fest, dass eine Datenfeldgruppe im Gruppenbereich (als Tab) angezeigt wird.
- **Werte**:
  - *true / false*: Ohne diese Annotation wird die Gruppe ohne Tab im Ticket-, Kunden- oder Ressourcenbereich angezeigt.

show-labels-in-edit

- **Typ**: layout
- **Beschreibung**: Ob die Datenfelder dieser Gruppe im Bearbeitungsmodus mit Labeln angezeigt werden sollen.
- **Werte**:
  - *true / false*: Seit Version 6.9.4.

show-labels-in-view

- **Typ**: layout
- **Beschreibung**: Ob die Datenfelder dieser Gruppe im Ansichtsmodus mit Labeln angezeigt werden sollen.
- **Werte**:
  - *true / false*: Seit Version 6.9.4.

show-tooltips

- **Typ**: layout
- **Beschreibung**: Ob die Datenfelder dieser Gruppe mit Tooltips angezeigt werden sollen.
- **Werte**:
  - *true / false*: Seit Version 6.9.4.

show-watermarks

- **Typ**: layout
- **Beschreibung**: Ob die Datenfelder dieser Gruppe mit Wasserzeichen angezeigt werden sollen.
- **Werte**:
  - *true / false*: Seit Version 6.9.4.

G.1.2.4 performance (Typ)

no-history

- **Typ**: performance
- **Beschreibung**: Legt fest, dass alle Datenfelder, die zu dieser Gruppe gehören, nicht protokolliert werden.
- **Werte**:
  - *true / false*: Legt fest, dass alle Datenfelder, die zu dieser Gruppe gehören, nicht protokolliert werden sollen. Mögliche Werte sind *true*, wenn die Annotation aktiv sein soll, oder *false*, was dem Entfernen der Annotation entspricht. Verwenden Sie diese Annotation, wenn Sie verhindern möchten, dass ein Protokoll für alle/viele Felder der Gruppe gespeichert wird. Wenn Sie nur die Protokollierung einzelner Felder verhindern möchten, können Sie die Annotation *no-history-field* auf Feldebene setzen.
    In CM-Versionen bis 6.10.2 wird auch die Übertragung der Feldhistorie an das DWH mit dieser Annotation gesteuert.
    Verwenden Sie dazu ab CM-Version 6.10.2 die Annotation *dwh-no-history*.

G.1.2.5 resource (Typ)

resource-fields-group-mode

- **Typ**: resource
- **Beschreibung**: Steuert den Modus einer Ressourcenfeldgruppe beim Editieren mit dem Web Client.
- **Werte**:
  - *internal / external*: Mögliche Werte: internal, external. Ein Ressourcengruppenfeld kann nicht im Web Client editiert werden, wenn der Wert *external* ist.
    Seit 6.10.4.0
    Entfernt in 6.10.5.0

resource-custom-fields-group-mode

- **Typ**: resource
- **Beschreibung**: Steuert den Modus einer Ressourcenfeldgruppe beim Editieren mit dem Web Client.
- **Werte**:
  - *internal / external*: Mögliche Werte: internal, external. Ein Ressourcengruppenfeld kann nicht im Web Client editiert werden, wenn der Wert *external* ist.
    Seit 6.10.5.0

G.1.2.6 ticket contact relation (Typ)

unit is a contact

- **Typ**: ticket contact relation
- **Beschreibung**: veraltet
- **Werte**:
    - *true/false*: Entfernt in Version 6.9.0.

G.2 System-Properties

Das folgende Kapitel enthält detaillierte Informationen über die in ConSol CM verwendeten System-Properties.

G.2.1 Alphabetische Liste der System-Properties

admin.email

- **Modul**: cmas-core-security
- **Beschreibung**: Die E-Mail-Adresse des ConSol CM-Administrators. Anfänglich wird hier der Wert genommen, den Sie bei der Systemeinrichtung eingegeben haben.
- **Typ**: string
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: myuser@consol.de
- **Seit**: 6.0

admin.login

- **Modul**: cmas-core-security
- **Beschreibung**: Der Name des ConSol CM-Administrators. Anfänglich wird hier der Wert genommen, den Sie bei der Systemeinrichtung eingegeben haben.
- **Typ**: string
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: admin
- **Seit**: 6.0

admin.tool.session.check.interval

- **Modul**: cmas-app-admin-tool
- **Beschreibung**: Intervall, in dem inaktive (beendete) Sitzungen im Admin Tool überprüft werden (in Sekunden).
- **Typ**: integer
- **Neustart erforderlich**: ja
- **System**: ja
- **Optional**: nein
- **Beispielwert**: 30
- **Seit**: 6.7.5

attachment.allowed.types

- **Modul**: cmas-core-server
- **Beschreibung**: Durch Kommas getrennte Liste der erlaubten Dateinamenserweiterungen (wenn kein Werte definiert ist, sind alle Dateinamenserweiterungen erlaubt).

- **Typ**: string
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: ja
- **Beispielwert**: txt,zip,doc
- **Seit**: 6.5.0

attachment.max.size

- **Modul**: cmas-core-server
- **Beschreibung**: Maximale Größe von Attachments in MB. Dies ist eine Validierungs-Property der CM-API. Sie steuert die Größe der Attachments in Tickets, Units und Ressourcen. Außerdem steuert sie die Größe der eingehenden (nicht ausgehenden!) E-Mail-Attachments im NIMH- sowie im Mule/ESB-Modus.
- **Typ**: integer
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: 100
- **Seit**: 6.4.0

attachment.upload.timeout

- **Modul**: cmweb-server-adapter
- **Beschreibung**: Definiert das Transaktions-Timeout in Minuten für das Hinzufügen von Attachments zu einem Ticket, einer Ressource oder einem Kunden. Dabei zählt die Zeit für den Upload aller Attachments einer Transaktion. Wenn das Timeout eintritt, werden alle Dateien, die temporär auf dem Server gespeichert waren, gelöscht. Es wird keine Datei hochgeladen.
- **Typ**: Integer
- **Neustart erforderliche**: nein
- **System**: ja
- **Optional**: ja
- **Beispielwert**: 3
- **Seit**: 6.10.5.3

authentication.method

- **Modul**: cmas-core-security
- **Description**: Methode der Bearbeiter-Authentifizierung für den Web Client (interne CM-Datenbank oder LDAP-Authentifizierung). Mögliche Werte sind LDAP oder DATABASE.
- **Typ**: string

- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: DATABASE
- **Seit**: 6.0

autocommit.cf.changes

- **Modul**: cmas-dwh-server
- **Beschreibung**: Definiert, ob DWH-Aufgaben, die aufgrund von Konfigurationsänderungen an Benutzerdefinierten Feldern anfallen, automatisch ohne manuelle Interaktion im Admin Tool ausgeführt werden. Diese Property kann auch im Admin Tool im Navigationselement *DWH* gesetzt werden. Der Standardwert und empfohlene Wert ist *false*.
- **Typ**: boolean
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: false
- **Seit**: 6.7.0

autocomplete.enabled

- **Modul**: cmas-app-admin-tool
- **Beschreibung**: Wenn diese System-Property fehlt oder der Wert *false* ist, wird das Navigationselement *Adress-Vervollständigung* im Admin Tool ausgeblendet.
- **Typ**: boolean
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: ja
- **Beispielwert**: true
- **Seit**: 6.9.2.0

automatic.booking.enabled

- **Modul**: cmweb-server-adapter
- **Description**: Wenn aktiviert, wird die Zeit für das Erstellen eines Kommentars oder einer E-Mail gemessen und eine automatische Zeitbuchung hinzugefügt.
- **Typ**: boolean
- **Neustart erforderlich**: nein
- **System**: ja

- **Optional**: ja
- **Beispielwert**: true
- **Seit**: 6.9.4.2

batch-commit-interval

- **Modul**: cmas-dwh-server
- **Beschreibung**: Anzahl der Objekte in einer JMS-Nachricht. Höhere Werte bedeuten eine bessere Übertragungsperformance und größeren Speicherverbrauch.
- **Typ**: integer
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: ja
- **Beispielwert**: 100
- **Seit**: 6.0.0

big.task.minimum.size

- **Modul**: cmas-core-index-common
- **Beschreibung**: Gibt die Minimalgröße eines Index-Tasks an (in Teilen, jeder Teil hat 100 Einheiten), um diesen Task als einen großen Task zu qualifizieren. Große Tasks haben eine niedrigere Priorität als normale.
- **Typ**: integer
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: 15 (Standardwert)
- **Seit**: 6.8.3

cache-cluster-name

- **Modul**: cmas-core-cache
- **Beschreibung**: Cache-Cluster-Name des JBoss.
- **Typ**: string
- **Neustart erforderlich**: ja
- **System**: ja
- **Optional**: nein
- **Beispielwert**: 635a6de1-629a-4129-8299-2d98633310f0
- **Seit**: 6.4.0

checkUserOnlineIntervalInSeconds

- **Modul**: cmweb-server-adapter
- **Beschreibung**: Das Intervall (in Sekunden), in dem geprüft wird, welche Benutzer online sind (Standard 180 Sekunden = 3 Minuten).
- **Typ**: integer
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: 180
- **Seit**: 6.0
- **Entfernt in**: 6.5 / 6.11.0.1

cluster.mode

- **Modul**: cmas-core-shared
- **Beschreibung**: Legt fest, ob CMAS in einem Cluster läuft.
- **Typ**: boolean
- **Neustart erforderlich**: ja
- **System**: ja
- **Optional**: nein
- **Beispielwert**: false
- **Seit**: 6.1.0

cmas.dropSchemaBeforeSetup

- **Modul**: cmas-setup-hibernate
- **Beschreibung**: Gibt an, ob das Schema während des Setups gelöscht werden soll (wurde).
- **Typ**: string
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: true
- **Seit**: 6.0

cmoffice.enabled

- **Modul**: cmweb-server-adapter
- **Beschreibung**: Legt fest, ob CM.Doc (vorher CM/Office) aktiviert ist.
- **Typ**: boolean

- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: false
- **Seit**: 6.4.0

cmoffice.strict.versioning.enabled

- **Modul**: cmweb-server-adapter
- **Beschreibung**: Steuert, ob der Speichervorgang für Microsoft Word- / OpenOffice-Dokumente ein neues Attachment erzeugt (*true*) oder das vorhandene Attachment überschreibt (*false*). Dies betrifft das Verhalten innerhalb einer Session mit dem Textbearbeitungsprogramm. Wenn das Programm beendet wird, funktioniert der Mechanismus zum Überschreiben nicht mehr.
- **Typ**: Boolean
- **Neustart erforderlich**: nein
- **System**: nein
- **Optional**: ja
- **Beispielwert**: true
- **Seit**: 6.10.5.4

commentRequiredForTicketCreation

- **Modul**: cmweb-server-adapter
- **Beschreibung**: Legt fest, ob der Kommentar ein Pflichtfeld für die Erstellung eines Tickets ist.
- **Typ**: boolean
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: true (Standardwert)
- **Seit**: 6.2.0

communication.channel

- **Modul**: cmas-dwh-server
- **Beschreibung**: Kommunikationskanal, mögliche Werte sind DIRECT (Datenbank-Kommunikationskanal, Standardwert seit 6.9.4.1) oder JMS (Standardwert vor 6.9.4.1). Diese System-Property muss vor 6.9.4.1 extra hinzugefügt werden.
- **Typ**: string
- **Neustart erforderlich**: nein
- **System**: ja

- **Optional**: nein
- **Beispielwert**: DIRECT
- **Seit**: 6.8.5.0
- **Entfernt in**: 6.11.0.0

config.data.version

- **Modul**: cmas-core-server
- **Beschreibung**: Die interne Versionsnummer der aktuellen Systemkonfiguration. Diese Property wird intern gepflegt. Ändern Sie sie nur, wenn Sie von ConSol dazu aufgefordert werden.
- **Typ**: integer
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: 11
- **Seit**: 6.0

contact.authentication.method

- **Modul**: cmas-core-security
- **Beschreibung**: Definiert die Kontakt-Authentifizierungsmethode für CM.Track, mögliche Werte sind DATABASE oder LDAP oder LDAP,DATABASE oder DATABASE,LDAP.
- **Typ**: string
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Seit**: 6.9.3.0

contact.inherit.permissions.only.to.own.customer.group

- **Modul**: cmas-core-security
- **Description**: Legt fest, ob der authentifizierte Kontakt in CM.Track alle Kundengruppen-Berechtigungen vom repräsentierenden Bearbeiter erbt (false) oder nur die Berechtigungen für die eigene Kundengruppe hat (true).
- **Typ**: boolean
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Seit**: 6.9.2.3

customizationVersion

- **Modul**: cmweb-server-adapter
- **Beschreibung**: UID der letzten Version der Web-Anpassung. Wird nur intern verwendet. Der Wert darf nicht geändert werden.
- **Typ**: string
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: cd58453e-f3cc-4538-8030-d15e8796a4a7
- **Seit**: 6.5.0

data.directory

- **Modul**: cmas-core-shared
- **Beschreibung**: Verzeichnis für die CMAS-Daten (z. B. Index)
- **Typ**: string
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: C:\Users\user\cmas
- **Seit**: 6.0

data.optimization

- **Modul**: cmweb-server-adapter
- **Beschreibung**: Definiert die Optimierung der Response-Daten. Bisher werden die folgenden Werte unterstützt (mehrere Werte können durch '|' getrennt gesetzt werden): MINIFICATION und COMPRESSION. MINIFICATION minimiert HTML-Daten, indem z. B. Leerzeichen und Kommentare entfernt werden. COMPRESSION wendet gzip-Komprimierung auf die HTTP-Response an. (Hinweis: Wenn das System im Cluster-Modus läuft und Sie parallel verschiedene Konfigurationen testen möchten, können Sie für jeden Cluster-Node verschiedene Werte setzen, indem Sie die System-Property nach dem Muster *data.optimization.nodeId* spezifizieren, um die Standard-Property zu überschreiben.)
- **Typ**: string
- **Neustart**: COMPRESSION kann ohne Neustart ein- und ausgeschaltet werden, für MINIFICATION ist ein Neustart erforderlich.
- **System**: ja
- **Optional**: ja
- **Beispielwert**: MINIFICATION|COMPRESSION

database.notification.enabled

- **Modul**: cmas-core-index-common
- **Beschreibung**: Legt fest, ob statt JMS der Database Notification Channel der Index-Aktualisierung verwendet werden soll.
- **Typ**: boolean
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: false
- **Seit**: 6.8.4.7

database.notification.redelivery.delay.seconds

- **Modul**: cmas-core-index-common
- **Beschreibung**: Legt bei Verwendung des Database Notification Channel der Index-Aktualisierung fest, mit welcher Verzögerung die Benachrichtigung erneut gesendet wird, wenn eine Exception auftritt.
- **Typ**: integer
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: 60
- **Seit**: 6.8.4.7

database.notification.redelivery.max.attempts

- **Modul**: cmas-core-index-common
- **Beschreibung**: Legt bei Verwendung des Database Notification Channel der Index-Aktualisierung fest, wie oft maximal versucht wird, die Benachrichtigung erneut zu senden, wenn eine Exception auftritt.
- **Typ**: integer
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: 60
- **Seit**: 6.8.4.7

defaultCommentClassName

- **Modul**: cmas-core-server
- **Beschreibung**: Standard-Textklasse für Kommentare.
- **Typ**: string
- **Neustart erforderlich**: nein
- **System**: nein
- **Optional**: ja
- **Beispielwert**:
- **Seit**: 6.3.0

defaultContentEntryClassName

- **Modul**: cmweb-server-adapter
- **Beschreibung**: Standard-Textklasse für neue ACIMs.
- **Typ**: string
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: default_class
- **Seit**: 6.3.0

defaultIncommingMailClassName

- **Modul**: cmas-core-server
- **Beschreibung**: Standard-Textklasse für eingehende E-Mails.
- **Typ**: string
- **Neustart erforderlich**: nein
- **System**: nein
- **Optional**: ja
- **Seit**: 6.3.0

defaultNumberOfCustomFieldsColumns

- **Modul**: cmweb-server-adapter
- **Beschreibung**: Standardanzahl an Spalten für Benutzerdefinierte Felder.
- **Typ**: integer
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein

- **Beispielwert**: 3
- **Seit**: 6.2.0

defaultOutgoingMailClassName

- **Modul**: cmas-core-server
- **Beschreibung**: Standard-Textklasse für ausgehende E-Mails.
- **Typ**: string
- **Neustart erforderlich**: nein
- **System**: nein
- **Optional**: ja
- **Beispielwert**:
- **Seit**: 6.3.0

delete.ticket.enabled

- **Modul**: cmas-app-admin-tool
- **Beschreibung**: Steuert, ob in der Ticketverwaltung im Admin Tool der Menüpunkt *Tickets löschen* im Kontextmenü der Ticketliste angezeigt wird.
- **Typ**: boolean
- **Neustart erforderlich**: nein
- **System**: nein
- **Optional**: ja
- **Beispielwert**: true
- **Seit**: 6.9.4.0

diffTrackingEnabled

- **Modul**: cmweb-server-adapter
- **Beschreibung**: Legt fest, ob das gleichzeitige Bearbeiten eines Tickets durch verschiedene Bearbeiter möglich ist. Der Standardwert ist *true*.
  false: Vorherige Art und Weise der Behandlung von Änderungen beim Editieren von Tickets. Wenn ein Ticket zwischendurch geändert wurde, kann der aktuelle Bearbeiter seine Änderungen nicht speichern, ohne die Seite vorher neu zu laden.
  true: Neuer Modus zur Behandlung von Änderungen. Wenn das Ticket geändert wurde, wird das Speichern von anderen Änderungen nicht mehr blockiert. Wenn der Teil des Tickets, der geändert wurde, genau der Teil ist, der vom speichernden Bearbeiter geändert wird, wird eine Informationsmeldung angezeigt, die Ticketänderungen werden aber trotzdem gespeichert.
- **Typ**: boolean
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein

- **Beispielwert**: true
- **Seit**: 6.10.1
- **Entfernt in**: 6.11.0

disable.admin.task.auto.commit

- **Modul**: cmas-core-index-common
- **Beschreibung**: Alle Tasks, die für eine Index-Aktualisierung erstellt werden, werden automatisch direkt nach ihrer Erstellung ausgeführt.
- **Typ**: boolean
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: false
- **Seit**: 6.6.1

dwh.mode

- **Modul**: cmas-dwh-server
- **Beschreibung**: Aktueller Modus der DWH-Datenübermittlung. Mögliche Werte sind OFF, ADMIN, LIVE.
- **Typ**: string
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: OFF
- **Seit**: 6.0.1

esb.directory

- **Modul**: cmas-esb-core
- **Beschreibung**: Von Mule/ESB verwendetes Verzeichnis.
- **Typ**: string
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: C:\Users\user\cmas\mule
- **Seit**: 6.0
- **Entfernt in**: 6.11.0

eviction.event.queue.size

- **Modul**: cmas-core-cache
- **Beschreibung**: Die Größe der Queue mit den Cache-Events. Der Standardwert ist 200000. Auf Systemen mit viel Traffic oder Last wird empfohlen, diesen Wert leicht zu erhöhen (bis 400000).
- **Typ**: integer
- **Neustart erforderlich**: ja
- **System**: ja
- **Optional**: nein
- **Beispielwert**: 200000
- **Seit**: 6.4.0

eviction.max.nodes

- **Modul**: cmas-core-cache
- **Beschreibung**: Setzt die maximale Größe der internen Caches. Der Standardwert ist 100000. Das Erhöhen dieses Wertes führt zu einem höheren Speicherverbrauch und wird nicht empfohlen, außer wenn ConSol explizit dazu auffordert.
- **Typ**: integer
- **Neustart erforderlich**: ja
- **System**: ja
- **Optional**: nein
- **Beispielwert**: 100000
- **Seit**: 6.4.0

eviction.wakeup.interval

- **Modul**: cmas-core-cache
- **Beschreibung**: Setzt das Intervall (in Millisekunden) zwischen zwei Verarbeitungszyklen von Cache-Queue-Events. Der Standardwert ist 3000. Auf Systemen mit viel Traffic oder Last wird empfohlen, den Wert zu verringern (Minimalwert ist 1500).
- **Typ**: integer
- **Neustart erforderlich**: ja
- **System**: ja
- **Optional**: nein
- **Beispielwert**: 3000
- **Seit**: 6.4.0

favoritesSizeLimit

- **Modul**: cmweb-server-adapter
- **Beschreibung**: Maximale Anzahl von Elementen in der Favoritenliste.
- **Typ**: integer
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: 10
- **Seit**: 6.0

fetchLock.interval

- **Modul**: cmas-workflow-jbpm
- **Beschreibung**:
- **Typ**: integer
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: 5000
- **Entfernt in**: 6.8.0

fetchSize.strategy

- **Modul**: cmas-core-server
- **Beschreibung**: Auswahl der Strategie für die Abholgröße von JDBC-Ergebnissatzes.
- **Typ**: string
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: ja
- **Beispielwert**: FetchSizePageBasedStrategy, FetchSizeThresholdStrategy, FetchSizeFixedStrategy
- **Seit**: 6.8.4.1

fetchSize.strategy.FetchSizeFixedStrategy.value

- **Modul**: cmas-core-server
- **Beschreibung**: Legt den Wert für Abholgrößen fest, wenn die ausgewählte Strategie für die Abholgröße *FetchSizeFixedStrategy* ist.
- **Typ**: integer
- **Neustart erforderlich**: nein

- **System**: ja
- **Optional**: ja
- **Beispielwert**: 150
- **Seit**: 6.8.4.1

fetchSize.strategy.FetchSizePageBasedStrategy.limit

- **Modul**: cmas-core-server
- **Beschreibung**: Legt den Wert für Abholgrößen fest, wenn die ausgewählte Strategie für die Abholgröße *FetchSizePageBasedStrategy* ist.
- **Typ**: integer
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: ja
- **Beispielwert**: 10000
- **Seit**: 6.8.4.1

fetchSize.strategy.FetchSizeThresholdStrategy.value

- **Modul**: cmas-core-server
- **Beschreibung**: Gibt Grenzwerte für Abholgrößen an, wenn die ausgewählte Strategie für die Abholgröße *FetchSizeThresholdStrategy* ist.
- **Typ**: integer
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: ja
- **Beispielwert**: 150,300,600,1000
- **Seit**: 6.8.4.1

filesystem.polling.threads.number

- **Modul**: cmas-nimh
- **Beschreibung**: Anzahl der Threads, die für die Abfrage der Datenbank-E-Mail-Warteschlange gestartet werden. Standardwert: 1.
- **Typ**: integer
- **Neustart erforderlich**: nein
- **System**: nein
- **Optional**: ja
- **Beispielwert**: 10
- **Seit**: 6.4.0

filesystem.polling.threads.shutdown.timeout.seconds

- **Modul**: cmas-nimh
- **Beschreibung**: Wartezeit nach dem Shutdown-Signal. Wenn der Timeout erreicht ist, wird der Thread beendet. Standardwert: 60.
- **Typ**: integer
- **Neustart erforderlich**: nein
- **System**: nein
- **Optional**: ja
- **Beispielwert**: 60
- **Seit**: 6.4.0

filesystem.polling.threads.watchdog.interval.seconds

- **Modul**: cmas-nimh
- **Beschreibung**: Intervall des Watchdog Threads. Standardwert: 30.
- **Typ**: integer
- **Neustart erforderlich**: nein
- **System**: nein
- **Optional**: ja
- **Beispielwert**: 60
- **Seit**: 6.4.0

filesystem.task.enabled

- **Modul**: cmas-nimh
- **Beschreibung**: Mit dieser System-Property kann der Service Thread eines bestimmten Pollers deaktiviert werden. Standardwert: true.
- **Typ**: boolean
- **Neustart erforderlich**: nein
- **System**: nein
- **Optional**: ja
- **Beispielwert**: true
- **Seit**: 6.4.0

filesystem.task.interval.seconds

- **Modul**: cmas-nimh
- **Beschreibung**: Standard-Intervall für den Abruf von Postfächern. Standardwert: 60 Sekunden
- **Typ**: integer
- **Neustart erforderlich**: nein

- **System**: nein
- **Optional**: ja
- **Beispielwert**: 60
- **Seit**: 6.4.0

filesystem.task.polling.folder

- **Modul**: cmas-nimh
- **Beschreibung**: Speicherort des Ordners, der überwacht und nach E-Mails im Format von eml-Dateien durchsucht wird. Standard: Unterverzeichnis "mail" des cmas-Datenverzeichnisses
- **Typ**: string
- **Neustart erforderlich**: nein
- **System**: nein
- **Optional**: ja
- **Beispielwert**: c://cmas//mail
- **Seit**: 6.4.0

filesystem.task.timeout.seconds

- **Modul**: cmas-nimh
- **Beschreibung**: Nach dieser Zeit (der Inaktivität) wird der Service Thread als beschädigt betrachtet und automatisch neu gestartet. Standardwert: 120 Sekunden
- **Typ**: integer
- **Neustart erforderlich**: nein
- **System**: nein
- **Optional**: ja
- **Beispielwert**: 60
- **Seit**: 6.4.0

filesystem.task.transaction.timeout.seconds

- **Modul**: cmas-nimh
- **Beschreibung**: Standard-Transaktions-Timeout für Transaktionen, die E-Mails abrufen. Sollte mit der Anzahl der Nachrichten, die gleichzeitig abgeholt werden, korrelieren. Standardwert: 60 Sekunden
- **Typ**: integer
- **Neustart erforderlich**: nein
- **System**: nein
- **Optional**: ja

- **Beispielwert**: 60
- **Seit**: 6.4.0

globalSearchResultSizeLimit

- **Modul**: cmweb-server-adapter
- **Beschreibung**: Maximale Anzahl an Ergebnissen in der Schnellsuche.
- **Typ**: integer
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: 10
- **Seit**: 6.0

helpFilePath

- **Modul**: cmweb-server-adapter
- **Beschreibung**: URL für die Onlinehilfe. Wenn der Wert nicht leer gelassen wird, wird der Hilfe-Button im Web Client angezeigt.
- **Typ**: string
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: ja
- **Beispielwert**: http://www.consol.de
- **Seit**: 6.2.1

hibernate.dialect

- **Modul**: cmas-setup-hibernate
- **Beschreibung**: Der Hibernate-Dialekt. Normalerweise wird dieser Wert während des initialen Setups gesetzt (abhängig vom Datenbanksystem).
- **Typ**: string
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: org.hibernate.dialect.MySQL5InnoDBDialect
- **Seit**: 6.0

hideTicketSubject

- **Modul**: cmweb-server-adapter
- **Beschreibung**: Wenn auf *true* gesetzt, wird das Ticketthema ausgeblendet.
- **Typ**: boolean
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: false
- **Seit**: 6.2.1

ignore-queues

- **Modul**: cmas-dwh-server
- **Beschreibung**: Durch eine durch Kommas getrennte Liste von Queue-Namen wird hier festgelegt, dass Tickets dieser Queues nicht ins DWH übermittelt werden.
- **Typ**: string
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: ja
- **Beispielwert**: QueueName1,QueueName2,QueueName3
- **Seit**: 6.6.19
- **Entfernt in**: 6.8.1

index.attachment

- **Modul**: cmas-core-index-common
- **Beschreibung**: Beschreibt, ob der Inhalt von Attachments indiziert wird.
- **Typ**: boolean
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: true
- **Seit**: 6.4.3

index.history

- **Modul**: cmas-core-index-common
- **Beschreibung**: Legt fest, ob das Unit- und das Ticketprotokoll indiziert werden.
- **Typ**: boolean

- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: false
- **Seit**: 6.1.0
- **Entfernt in**: 6.11.0

index.status

- **Modul**: cmas-core-index-common
- **Beschreibung**: Status des Indexers, mögliche Werte sind RED, YELLOW, GREEN, werden im Admin Tool angezeigt.
- **Typ**: string
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: GREEN
- **Seit**: 6.6.1

index.task.worker.threads

- **Modul**: cmas-core-index-common
- **Beschreibung**: Legt fest, wie viele Threads benutzt werden, um Index-Aufgaben auszuführen (Synchronisierung, Administrations- und Reparatur-Aufgaben).
- **Typ**: integer
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: 1 (Standardwert) (wir empfehlen einen Wert zu verwenden, der nicht größer als 2 ist)
- **Seit**: 6.6.14, 6.7.3. Seit 6.8.0 und auschließlich in 6.6.21 sind auch normale (live) Index-Aktualisierungen von dieser Property betroffen.

index.version.current

- **Modul**: cmas-core-index-common
- **Beschreibung**: Enthält Informationen über die derzeitige (möglicherweise veraltete) Index-version.
- **Typ**: integer
- **Neustart erforderlich**: nein

- **System**: ja
- **Optional**: nein
- **Beispielwert**: 1 (Standardwert)
- **Seit**: 6.7.0

index.version.newest

- **Modul**: cmas-core-index-common
- **Beschreibung**: Enthält Informationen, welche Indexversion als die neueste betrachtet wird.
- **Typ**: integer
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: 1 (Standardwert)
- **Seit**: 6.7.0

indexed.assets.per.thread.in.memory

- **Modul**: cmas-core-index-common
- **Beschreibung**: Legt fest, wie viele Assets während des Indizierens pro Thread auf einmal in den Speicher geladen werden.
- **Typ**: integer
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: 200 (Standardwert)
- **Seit**: 6.8.0

indexed.engineers.per.thread.in.memory

- **Modul**: cmas-core-index-common
- **Beschreibung**: Legt fest, wie viele Bearbeiter während des Indizierens pro Thread auf einmal in den Speicher geladen werden.
- **Typ**: integer
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: 300 (Standardwert)
- **Seit**: 6.6.14, 6.7.3

indexed.resources.per.thread.in.memory

- **Modul**: cmas-core-index-common
- **Beschreibung**: Legt fest, wie viele Ressourcen während des Indizierens pro Thread auf einmal in den Speicher geladen werden.
- **Typ**: integer
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: 200 (Standardwert)
- **Seit**: 6.10.0.0

indexed.tickets.per.thread.in.memory

- **Modul**: cmas-core-index-common
- **Beschreibung**: Legt fest, wie viele Tickets während des Indizierens pro Thread auf einmal in den Speicher geladen werden.
- **Typ**: integer
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: 100 (Standardwert)
- **Seit**: 6.6.14, 6.7.3

indexed.units.per.thread.in.memory

- **Modul**: cmas-core-index-common
- **Beschreibung**: Legt fest, wie viele Units während des Indizierens pro Thread auf einmal in den Speicher geladen werden.
- **Typ**: integer
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: 200 (Standardwert)
- **Seit**: 6.6.14, 6.7.3

initialized

- **Modul**: cmas-setup-manager
- **Beschreibung**: Legt fest, ob CMAS initialisiert ist. Wenn dieser Wert fehlt oder nicht *auf* true gesetzt ist, wird das Setup ausgeführt.

- **Typ**: boolean
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: true
- **Seit**: 6.0

> ⚠ Seien Sie mit der Verwendung dieser System-Property sehr vorsichtig! Wenn Sie den Wert auf *false* setzen, wird der ConSol CM-Server beim nächsten Systemstart das System-Setup ausführen, d. h. alle Daten des bestehenden Systems gehen verloren, einschließlich der System-Properties!

is.cmrf.alive

- **Modul**: cmas-dwh-server
- **Beschreibung**: Als Startpunkt sollte die Zeit genommen werden, bei der zuletzt eine Nachricht an das CMRF gesendet wurde. Wenn nach diesem Wert (in Sekunden) keine Antwort vom CMRF empfangen wird, wird ein DWH-Betriebsstatus mit der Fehlermeldung, dass das CMRF nicht erreichbar ist, erstellt.
- **Typ**: integer
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: 1200
- **Seit**: 6.7.0

java.naming.factory.initial

- **Modul**: cmas-dwh-server
- **Beschreibung**: Factory-Java-Klasse für DWH context factory.
- **Typ**: string
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: org.jnp.interfaces.NamingContextFactory
- **Seit**: 6.0.1
- **Entfernt in**: 6.11.0.0

java.naming.factory.url.pkgs

- **Modul**: cmas-dwh-server
- **Beschreibung**:
- **Typ**: string
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: org.jboss.naming:org.jnp.interfaces
- **Seit**: 6.0.1
- **Entfernt in**: 6.11.0.0

java.naming.provider.url

- **Modul**: cmas-dwh-server
- **Beschreibung**: URL des Naming Provider.
- **Typ**: string
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: localhost
- **Seit**: 6.0.1
- **Entfernt in**: 6.11.0.0

jobExecutor.adminMail

- **Modul**: cmas-workflow-engine
- **Beschreibung**: E-Mail-Adresse, an die Benachrichtigungs-E-Mails, die Probleme der Job-ausführung betreffen, geschickt werden (wenn die Anzahl der Neuversuche überschritten wurde).
- **Typ**: string
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: ja
- **Beispielwert**: admin@consol.de
- **Seit**: 6.8.0

jobExecutor.idleInterval

- **Modul**: cmas-workflow-jbpm
- **Beschreibung**:

- **Typ**: integer
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: 45000
- **Entfernt in**: 6.8.0
- **Ersetzt durch**: jobExecutor.idleInterval.seconds

jobExecutor.idleInterval.seconds

- **Modul**: cmas-workflow-engine
- **Beschreibung**: Legt fest, wie oft der Job Executor Thread nach neuen Jobs zum Ausführen sucht.
- **Typ**: integer
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: ja
- **Beispielwert**: 45 (Standardwert bis CM-Version 6.10.5.2. Der Standardwert für CM-Versionen 6.10.5.3 und höher ist 5)
- **Seit**: 6.8.0

jobExecutor.jobExecuteRetryNumber

- **Modul**: cmas-workflow-jbpm
- **Beschreibung**:
- **Typ**: integer
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: 5
- **Entfernt in**: 6.8.0
- **Ersetzt durch**: jobExecutor.jobMaxRetries

jobExecutor.jobMaxRetries

- **Modul**: cmas-workflow-engine
- **Beschreibung**: Steuert die Anzahl der erneuten Versuche, die der Job Executor unternimmt, bevor er einen Job als fehlgeschlagen deklariert.
- **Typ**: integer
- **Neustart erforderlich**: nein

- **System**: ja
- **Optional**: ja
- **Beispielwert**: 5 (Standardwert)
- **Seit**: 6.8.0

jobExecutor.jobMaxRetriesReachedSubject

- **Modul**: cmas-workflow-engine
- **Beschreibung**: Der Betreff für Benachrichtigungs-E-Mails, die Administratoren über fehlgeschlagene Job-Ausführungen erhalten.
- **Typ**: string
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: ja
- **Beispielwert**: Maximale Anzahl der Neuversuche für Job erreicht. Job wurde entfernt! (Standard)
- **Seit**: 6.8.0

jobExecutor.lockingLimit

- **Modul**: cmas-workflow-engine
- **Beschreibung**: Anzahl der gleichzeitig gelockten (als "in der Ausführung" markierten) Jobs des Job Executor Threads.
- **Typ**: integer
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: ja
- **Beispielwert**: 5 (Standardwert seit CM-Version 6.10.5.3)
- **Seit**: 6.8.0

jobExecutor.lockTimeout.seconds

- **Modul**: cmas-workflow-engine
- **Beschreibung**: Legt fest, wie lange ein Job vom Job Executor (als "in der Ausführung" markiert) gelockt werden kann.
- **Typ**: integer
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: ja

- **Beispielwert**: 360 (Standardwert)
- **Seit**: 6.8.0

jobExecutor.mailFrom

- **Modul**: cmas-workflow-engine
- **Beschreibung**: E-Mail-Adresse, die als From-Header für Admin-Benachrichtigungen eingesetzt wird.
- **Typ**: string
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: ja
- **Beispielwert**: jobexecutor@consol.de
- **Seit**: 6.8.0

jobExecutor.maxInactivityInterval.minutes

- **Modul**: cmas-workflow-engine
- **Beschreibung**: Länge der erlaubten Inaktivität des Job Executors in Minuten (z. B. wenn er durch eine Langzeit-Ausführung gesperrt wird). Nach dieser Zeit werden die Executor Threads neu gestartet.
- **Typ**: integer
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: ja Standardwert ist auf 30 Minuten gesetzt.
- **Beispielwert**: 15 (Standardwert)
- **Seit**: 6.9.2.0

jobExecutor.threads

- **Modul**: cmas-workflow-engine
- **Beschreibung**: Anzahl der Job Executor Threads.
- **Typ**: integer
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: ja
- **Beispielwert**: 1 (Standardwert)
- **Seit**: 6.8.0

jobExecutor.timerRetryInterval

- **Modul**: cmas-workflow-jbpm
- **Beschreibung**:
- **Typ**: integer
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: 10000
- **Entfernt in**: 6.8.0
- **Ersetzt durch**: jobExecutor.timerRetryInterval.seconds

jobExecutor.timerRetryInterval.seconds

- **Modul**: cmas-workflow-engine
- **Beschreibung**: Legt fest, wie lange der Job Executor Thread nach einem Fehler bei der Job-Ausführung wartet.
- **Typ**: integer
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: ja
- **Beispielwert**: 10 (Standardwert bis CM-Version 6.10.5.2. Der Standardwert für CM-Versionen 6.10.5.3 und höher ist 30.)
- **Seit**: 6.8.0

jobExecutor.txTimeout.seconds

- **Modul**: cmas-workflow-engine
- **Beschreibung**: Transaktions-Timeout für die Job-Ausführung.
- **Typ**: integer
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: ja
- **Beispielwert**: 60 (Standardwert)
- **Seit**: 6.8.0

kerberos.v5.enabled

- **Modul**: cmas-core-security
- **Description**: Legt fest, ob SSO über Kerberos aktiviert ist.
- **Typ**: boolean

- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: false (Standardwert, wenn Kerberos bei der Systemeinrichtung nicht aktiviert wurde)
- **Seit**: 6.2.0

kerberos.v5.username.regex

- **Modul**: cmas-core-security
- **Beschreibung**: Regulärer Ausdruck für die Zuordnung des Kerberos Principals zum Login des CM-Bearbeiters
- **Typ**: string
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: (.*)@.*
- **Seit**: 6.2.0

last.config.change

- **Modul**: cmas-core-server
- **Beschreibung**: Zufällige UUID, die während der letzten Konfigurationsänderung generiert wurde.
- **Typ**: string
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: 2573c7b7-2bf5-47ff-b5a2-bad31951a266
- **Seit**: 6.1.0, 6.2.1

ldap.authentication

- **Modul**: cmas-core-security
- **Description**: Verwendete Authentifizierungsmethode, wenn LDAP-Authentifizierung benutzt wird.
- **Typ**: string
- **Neustart erforderlich**: ja
- **System**: ja
- **Optional**: nein

- **Beispielwert**: simple
- **Seit**: 6.0

ldap.basedn

- **Modul**: cmas-core-security
- **Beschreibung**: Base DN für die Suche von LDAP-Benutzerkonten, wenn LDAP-Authentifizierung verwendet wird.
- **Typ**: string
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: ou=accounts,dc=consol,dc=de
- **Seit**: 6.0

ldap.certificate.basedn

- **Modul**: cmas-core-server
- **Beschreibung**: Base DN für den Speicherort der Zertifikate im LDAP-Verzeichnisbaum. Wenn nicht angegeben, wird *cmas-core-security, ldap.basedn* verwendet.
- **Typ**: string
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: ja
- **Beispielwert**: ou=accounts,dc=consol,dc=de
- **Seit**: 6.8.4

ldap.certificate.content.attribute

- **Modul**: cmas-core-server
- **Description**: Name des LDAP-Attributs, das angibt, wo Zertifikatsdaten im LDAP-Verzeichnisbaum gespeichert werden. Standardwert: usercertificate
- **Typ**: string
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: ja
- **Beispielwert**: usercertificate
- **Seit**: 6.8.4

ldap.certificate.password

- **Modul**: cmas-core-server
- **Beschreibung**: Passwort des LDAP-Zertifikatmanagers. Wenn nicht gesetzt, wird *cmas-core-security, ldap.password* verwendet.
- **Typ**: string
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: ja
- **Seit**: 6.8.4

ldap.certificate.providerurl

- **Modul**: cmas-core-server
- **Beschreibung**: URL des LDAP-Zertifikat-Providers. Wenn nicht gesetzt, wird *cmas-core-security, ldap.providerurl* verwendet.
- **Typ**: string
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: ja
- **Beispielwert**: ldap://ldap.consol.de:389
- **Seit**: 6.8.4

ldap.certificate.searchattr

- **Modul**: cmas-core-server
- **Description**: Name des LDAP-Attributs, mit dem Zertifikate im LDAP-Verzeichnisbaum gesucht werden. Standardwert: mail
- **Typ**: string
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: ja
- **Beispielwert**: mail
- **Seit**: 6.8.4

ldap.certificate.userdn

- **Modul**: cmas-core-server
- **Beschreibung**: DN des LDAP-Zertifikatmanagers. Wenn nicht gesetzt, wird *cmas-core-security, ldap.userdn* verwendet.
- **Typ**: string

- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: ja
- **Seit**: 6.8.4

ldap.contact.name.basedn

- **Modul**: cmas-core-security
- **Description**: Base DN für die Suche nach der Kontakt-DN mittels LDAP-ID (z. B. ou=a-ccounts,dc=consol,dc=de).
- **Typ**: string
- **Neustart erforderlich**: nein
- **System**: nein
- **Optional**: ja
- **Seit**: 6.9.3.0

ldap.contact.name.password

- **Modul**: cmas-core-security
- **Beschreibung**: Passwort für die Suche nach der Kontakt-DN mittels LDAP-ID. Wenn nicht gesetzt, wird ein anonymes Konto verwendet.
- **Typ**: string
- **Neustart erforderlich**: nein
- **System**: nein
- **Optional**: ja
- **Seit**: 6.9.3.0

ldap.contact.name.providerurl

- **Modul**: cmas-core-security
- **Beschreibung**: Adresse des LDAP-Servers (ldap[s]://host:port).
- **Typ**: string
- **Neustart erforderlich**: nein
- **System**: nein
- **Optional**: ja
- **Seit**: 6.9.3.0

ldap.contact.name.searchattr

- **Modul**: cmas-core-security
- **Beschreibung**: Attribut für die Suche nach der Kontakt-DN mittels LDAP-ID (z. B. uid).

- **Typ**: string
- **Neustart erforderlich**: nein
- **System**: nein
- **Optional**: ja
- **Seit**: 6.9.3.0

ldap.contact.name.userdn

- **Modul**: cmas-core-security
- **Beschreibung**: Benutzer-DN für die Suche nach Kontakt-DN mittels LDAP-ID. Wenn nicht gesetzt, wird ein anonymes Konto verwendet.
- **Typ**: string
- **Neustart erforderlich**: nein
- **System**: nein
- **Optional**: ja
- **Seit**: 6.9.3.0

ldap.initialcontextfactory

- **Modul**: cmas-core-security
- **Beschreibung**: Name der Java-Klasse für die Initial Context Factory der LDAP-Implementierung bei der Verwendung von LDAP-Authentifizierung. Ist üblicherweise *com.sun.jndi.ldap.LdapCtxFactory*.
- **Typ**: string
- **Neustart erforderlich**: ja
- **System**: ja
- **Optional**: nein
- **Beispielwert**: com.sun.jndi.ldap.LdapCtxFactory
- **Seit**: 6.0

ldap.password

- **Modul**: cmas-core-security
- **Beschreibung**: Passwort für die Verbindung zum LDAP, um Benutzer zu suchen, wenn LDAP-Authentifizierung verwendet wird. Wird nur benötigt, wenn die Suche nicht anonym ausgeführt werden kann.
- **Typ**: password
- **Neustart erforderlich**: nein
- **System**: ja

- **Optional**: ja
- **Seit**: 6.1.2

ldap.providerurl

- **Modul**: cmas-core-security
- **Beschreibung**: LDAP-Provider, wenn LDAP-Authentifizierung verwendet wird.
- **Typ**: string
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: ldap://myserver.consol.de:389
- **Seit**: 6.0

ldap.searchattr

- **Modul**: cmas-core-security
- **Beschreibung**: Suchattribut für die Suche nach LDAP-Einträgen, die mit dem CM-Login verbunden sind.
- **Typ**: string
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: uid
- **Seit**: 6.0

ldap.userdn

- **Modul**: cmas-core-security
- **Beschreibung**: LDAP-Benutzer für die Verbindung zum LDAP, um Benutzer zu suchen, wenn LDAP-Authentifizierung verwendet wird. Wird nur benötigt, wenn die Suche nicht anonym ausgeführt werden kann.
- **Typ**: string
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: ja
- **Seit**: 6.1.2

mail.attachments.validation.info.sender

- **Modul**: cmas-esb-mail
- **Beschreibung**: Setzt den From-Header bei Attachments vom Typ *error notification mail*. Standardmäßig wird die E-Mail-Adresse des Administrators verwendet, die bei der Systeminstallation angegeben wurde.
- **Typ**: string
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: admin@consolcm.com
- **Seit**: 6.7.5
- **Entfernt in**: 6.11.0

mail.attachments.validation.info.sender

- **Modul**: cmas-nimh-extension
- **Beschreibung**: Setzt den From-Header bei Attachments vom Typ *error notification mail*.
- **Typ**: string
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: admin@mail.com
- **Seit**: 6.7.5

> Diese System-Property entspricht der alten *cmas-esb-mail, mail.attachments.validation.info.sender*

mail.attachments.validation.info.subject

- **Modul**: cmas-esb-mail
- **Beschreibung**: Setzt den Betreff bei Attachments vom Typ *error notification mail*.
- **Typ**: string
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: E-Mail konnte nicht verarbeitet werden, weil ihre Attachments zurückgewiesen wurden!

- **Seit**: 6.7.5
- **Entfernt in**: 6.11.0

mail.attachments.validation.info.subject

- **Modul**: cmas-nimh-extension
- **Beschreibung**: Setzt den Betreff bei Attachments vom Typ error notification mail.
- **Typ**: string
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: E-Mail konnte nicht verarbeitet werden, weil ihre Attachments zurückgewiesen wurden!
- **Seit**: 6.7.5

> Diese System-Property entspricht der alten *cmas-esb-mail, mail.attachments.validation.info.subject*

mail.callname.pattern

- **Modul**: cmas-esb-mail
- **Beschreibung**: Regulärer Ausdruck für den Betreff von eingehenden E-Mails. Verfügbar als TICKET_NAME_PATTERN_FORMAT in Skripten für eingehende E-Mails.
- **Typ**: string
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: .*?Ticket\s+\((\S+)\).*
- **Seit**: 6.0
- **Entfernt in**: 6.11.0

mail.cluster.node.id

- **Modul**: cmas-esb-mail
- **Beschreibung**: Nur der Node, dessen *mail.cluster.node.id* gleich *cmas.clusternode.id* ist, startet den Mule/ESB E-Mail-Service.
- **Typ**: string
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein

- **Beispielwert**: unspecified
- **Seit**: 6.6.5
- **Entfernt in**: 6.11.0

mail.db.archive

- **Modul**: cmas-esb-mail
- **Beschreibung**: Wenn dieser Wert auf *true* gesetzt ist, werden eingehende E-Mails in der Datenbank archiviert.
- **Typ**: boolean
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: ja
- **Beispielwert**: false (Standardwert)
- **Seit**: 6.8.5.5
- **Entfernt in**: 6.11.0

Obsolet! Im Mule/ESB-Modus werden keine E-Mails in der Datenbank gespeichert. E-Mails, die nicht verarbeitet werden können, werden im Dateisystem gespeichert, siehe Abschnitt *E-Mail-Backups* im *ConSol CM Administratorhandbuch*.

mail.db.archive

- **Modul**: cmas-nimh-extension
- **Beschreibung**: Wenn dieser Wert auf *true* gesetzt ist, werden eingehende E-Mails in der Datenbank archiviert.
- **Typ**: boolean
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: ja
- **Beispielwert**: false (Standardwert)
- **Seit**: 6.8.5.5

mail.delete.read

- **Modul**: cmas-esb-mail
- **Beschreibung**: Legt fest, ob CM die per IMAP(S) abgeholten E-Mails löscht. Wenn der Wert auf *true* gesetzt wird, werden die E-Mails nach der Abholung gelöscht. Standardmäßig werden die per IMAP(S) abgeholten E-Mails nicht gelöscht. Hinweis: E-Mails, die per POP3(S) abgeholt werden, werden immer gelöscht.
- **Typ**: boolean
- **Neustart erforderlich**: nein

- **System**: ja
- **Optional**: nein
- **Beispielwert**: true
- **Seit**: 6.7.3
- **Entfernt in**: 6.11.0

mail.encryption

- **Modul**: cmas-core-server
- **Beschreibung**: Wenn dieser Wert auf *true* gesetzt ist, ist im Ticket-E-Mail-Editor die Checkbox zur Verschlüsselung der E-Mail standardmäßig aktiviert.
- **Typ**: boolean
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: true (Standardwert = false)
- **Seit**: 6.8.4.0

mail.error.from.address

Diese System-Property entspricht der alten *cmas-esb-mail,mail.mule.service*

mail.error.to.address

Diese System-Property entspricht der alten *cmas-esb-mail, mail.process.error*

mail.from

- **Modul**: cmweb-server-adapter
- **Beschreibung**: Diese E-Mail-Adresse wird anstelle der E-Mail-Adresse des Bearbeiters in E-Mail-Konversationen verwendet.
- **Typ**: string
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: ja
- **Seit**: 6.1.2

mail.incoming.uri

- **Modul**: cmas-esb-mail
- **Beschreibung**: URL für eingehende E-Mails.

- **Typ**: string
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: pop3://cm-incoming-user:password@localhost:10110
- **Seit**: 6.0
- **Entfernt in**: 6.11.0

Dieser Wert sollte nicht innerhalb des System-Property-Fensters verändert werden. Die Postfächer sollten stattdessen im Admin Tool im Navigationselement *E-Mail* konfiguriert werden. Wenn Sie dieses Element für die Konfiguration benutzen, können Sie alle Einträge kontrolliert konfigurieren, d. h. für jedes Postfach, das hinzugefügt wird, baut CM während der Einrichtung eine Testverbindung auf. Auf diese Weise ist es nicht möglich, falsche Werte einzugeben.

mail.max.restarts

- **Modul**: cmas-esb-mail
- **Beschreibung**: Maximale Anzahl der Neustarts des E-Mail-Services, bevor aufgegeben wird.
- **Typ**: integer
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: 3
- **Seit**: 6.0
- **Entfernt in**: 6.11.0

mail.mime.strict

- **Modul**: cmas-esb-mail
- **Beschreibung**: Wenn dieser Wert auf *false* gesetzt ist, werden E-Mail-Adressen nicht auf strikte MIME-Übereinstimmung geparst. Standard ist *true*, was bedeutet, dass auf strikte MIME-Übereinstimmung geprüft wird.
- **Typ**: boolean
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: false

- **Seit**: 6.6.17, 6.7.3
- **Entfernt in**: 6.11.0

mail.mule.service

- **Modul**: cmas-esb-mail
- **Beschreibung**: From-Adresse für E-Mails, die vom Mule-Service aus gesendet werden.
- **Typ**: email
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: myuser@consol.de
- **Seit**: 6.0
- **Entfernt in**: 6.11.0

mail.notification.engineerChange

- **Modul**: cmas-core-server
- **Description**: Legt fest, ob eine Benachrichtigungs-E-Mail verschickt wird, wenn der Bearbeiter eines Tickets wechselt.
- **Typ**: boolean
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: true
- **Seit**: 6.1.0

mail.notification.sender

- **Modul**: cmas-core-server
- **Beschreibung**: From-Adresse der Benachrichtigungs-E-Mails, die verschickt werden, wenn der Bearbeiter eines Tickets wechselt. Wenn nicht gesetzt, wird stattdessen *cmas-core-security, admin.email* verwendet.
- **Typ**: string
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: ja
- **Beispielwert**: cm6notification@cm6installation
- **Seit**: 6.6.3

mail.on.error

- **Modul**: cmas-nimh-extension
- **Beschreibung**: Wenn diese Property auf *true* gesetzt ist, wird im Fall, dass eine E-Mail nicht verarbeitet werden konnte, ein Fehler-E-Mail an die oben konfigurierten E-Mail-Adressen gesendet. Standardwert: true.
- **Typ**: boolean
- **Neustart erforderlich**: nein
- **System**: nein
- **Optional**: ja
- **Beispielwert**: false
- **Seit**: 6.4.0

mail.polling.interval

- **Modul**: cmas-esb-mail
- **Beschreibung**: Abrufintervall für E-Mails in Millisekunden.
- **Typ**: integer
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: 60000
- **Seit**: 6.0
- **Entfernt in**: 6.11.0

mail.process.error

- **Modul**: cmas-esb-mail
- **Beschreibung**: To-Adresse für E-Mails mit Fehlermeldungen des Mule. Standardmäßig wird die E-Mail-Adresse des Administrators verwendet, die bei der Systeminstallation angegeben wurde.
- **Typ**: email
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: myuser@consol.de
- **Seit**: 6.0
- **Entfernt in**: 6.11.0

mail.process.error

- **Modul**: cmas-nimh-extension
- **Beschreibung**: To-Adresse für E-Mails mit Fehlermeldungen des Mule.
- **Typ**: email
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: myuser@consol.de
- **Seit**: 6.4.0

mail.process.retry.attempts

- **Modul**: cmas-esb-mail
- **Beschreibung**: Anzahl der erneuten Versuche, wenn E-Mails verarbeitet werden.
- **Typ**: integer
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: 3
- **Seit**: 6.0.2
- **Entfernt in**: 6.11.0

mail.process.timeout

- **Modul**: cmas-esb-mail
- **Beschreibung**: Timeout für die E-Mail-Verarbeitung in Sekunden.
- **Typ**: integer
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: 60
- **Seit**: 6.1.3
- **Entfernt in**: 6.11.0

mail.redelivery.retry.count

- **Modul**: cmas-esb-mail
- **Beschreibung**: Gibt die Anzahl der Versuche an, eine E-Mail aus dem CM-System erneut zuzustellen.

- **Typ**: integer
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: 3
- **Seit**: 6.1.0
- **Entfernt in**: 6.11.0

mail.reply.to

- **Modul**: cmweb-server-adapter
- **Beschreibung**: Wenn dieser Wert gesetzt ist, zeigt der Web Client diesen Wert beim Versenden einer E-Mail im Reply-To-Feld an.
- **Typ**: string
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: ja
- **Seit**: 6.0.1

> ⚠ Bitte lesen Sie dazu die ausführlichen Informationen über Reply-To-Adressen in ConSol CM im Abschnitt *Skripte des Typs E-Mail* im *ConSol CM Administratorhandbuch*.

mail.sender.address

- **Modul**: cmas-workflow-jbpm
- **Beschreibung**: From-Adresse für E-Mails, die von der Workflow-Engine versendet werden.
- **Typ**: string
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: myuser@consol.de
- **Entfernt in**: 6.8.0
- **Ersetzt durch**: jobExecutor.mailFrom

mail.smtp.email

- **Modul**: cmas-core-server
- **Beschreibung**: SMTP-E-Mail-URL für ausgehende E-Mails.
- **Typ**: string

- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: smtp://mail.mydomain.com:25
- **Seit**: 6.0

mail.smtp.envelopesender

- **Modul**: cmas-core-server
- **Beschreibung**: E-Mail-Adresse, die als Absender im SMTP-Envelope benutzt wird. Wenn nicht gesetzt, wird die From-Adresse der E-Mail benutzt.
- **Typ**: string
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: mysender@mydomain.com
- **Seit**: 6.5.7

mail.ticketname.pattern

- **Modul**: cmas-nimh-extension
- **Beschreibung**: Regulärer Ausdruck, mit dem der Ticketname im Betreff eingehender E-Mails identifiziert wird.
- **Typ**: string
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: .*?Ticket\s+\((\S+)\).*
- **Seit**: 6.4.0

mailbox.1.connection.host

- **Modul**: cmas-nimh
- **Beschreibung**: Host (Server) für das erste konfigurierte Postfach. Überschreibt den Standardparameter *mailbox.default.connection.host*.

mailbox.1.connection.password

- **Modul**: cmas-nimh
- **Beschreibung**: Passwort für das erste konfigurierte Postfach. Überschreibt den Standardparameter *mailbox.default.connection.password*.

mailbox.1.connection.port

- **Modul**: cmas-nimh
- **Beschreibung**: Port für das erste konfigurierte Postfach. Überschreibt den Standardparameter *mailbox.default.connection.port*.

mailbox.1.connection.protocol

- **Modul**: cmas-nimh
- **Beschreibung**: Protokoll (z. B. IMAP oder POP3) für das erste konfigurierte Postfach. Überschreibt den Standardparameter *mailbox.default.connection.protocol*.

mailbox.1.connection.username

- **Modul**: cmas-nimh
- **Beschreibung**: Benutzername für das erste konfigurierte Postfach. Überschreibt den Standardparameter *mailbox.default.connection.username*.

mailbox.2.connection.host

- **Modul**: cmas-nimh
- **Beschreibung**: Host (Server) für das zweite konfigurierte Postfach. Überschreibt den Standardparameter *mailbox.default.connection.host*.

mailbox.2.connection.password

- **Modul**: cmas-nimh
- **Beschreibung**: Passwort für das zweite konfigurierte Postfach. Überschreibt den Standardparameter *mailbox.default.connection.password*.

mailbox.2.connection.port

- **Modul**: cmas-nimh
- **Beschreibung**: Port für das zweite konfigurierte Postfach. Überschreibt den Standardparameter *mailbox.default.connection.port*.

mailbox.2.connection.protocol

- **Modul**: cmas-nimh
- **Beschreibung**: Protokoll (z. B. IMAP oder POP3) für das zweite konfigurierte Postfach. Überschreibt den Standardparameter *mailbox.default.connection.protocol*.

mailbox.2.connection.username

- **Modul**: cmas-nimh
- **Beschreibung**: Benutzername für das zweite konfigurierte Postfach. Überschreibt den Standardparameter *mailbox.default.connection.username*.

Für alle Postfach-Properties im Zusammenhang mit NIMH wird folgendes Prinzip eingesetzt: Es wird eine Standard-Property definiert (z.B *mailbox.default.connection.port*). Wenn keine postfachspezifische Property konfiguriert ist, wird dieser Standardwert verwendet.

mailbox.default.connection.host

- **Modul**: cmas-nimh
- **Beschreibung**: Host (Servername) eines bestimmten Postfachs, aus dem der Poller E-Mails liest.
- **Typ**: string
- **Neustart erforderlich**: nein
- **System**: nein
- **Optional**: ja
- **Beispielwert**: 10.10.1.157
- **Seit**: 6.4.0

mailbox.default.connection.password

- **Modul**: cmas-nimh
- **Beschreibung**: Passwort für ein bestimmtes Postfach, aus dem der Poller E-Mails liest.
- **Typ**: string
- **Neustart erforderlich**: nein
- **System**: nein
- **Optional**: ja
- **Beispielwert**: consol
- **Seit**: 6.4.0

mailbox.default.connection.port

- **Modul**: cmas-nimh
- **Beschreibung**: Port für ein bestimmtes Postfach, aus dem der Poller E-Mails liest.
- **Typ**: string
- **Neustart erforderlich**: nein
- **System**: nein
- **Optional**: ja
- **Beispielwert**: 143
- **Seit**: 6.4.0

mailbox.default.connection.protocol

- **Modul**: cmas-nimh
- **Beschreibung**: Protokoll des Pollers, z. B. IMAP oder POP3. Kein Standardwert.
- **Typ**: string
- **Neustart erforderlich**: nein
- **System**: nein
- **Optional**: ja
- **Beispielwert**: imap
- **Seit**: 6.4.0

mailbox.default.connection.username

- **Modul**: cmas-nimh
- **Beschreibung**: Benutzername für ein bestimmtes Postfach, aus dem der Poller E-Mails liest.
- **Typ**: string
- **Neustart erforderlich**: nein
- **System**: nein
- **Optional**: ja
- **Beispielwert**: username
- **Seit**: 6.4.0

mailbox.default.session.mail.debug

- **Modul**: cmas-nimh
- **Beschreibung**: Beispiel für javax.mail-Property - ermöglicht detaillierteres Session-Debugging mit javax.mail.
- **Typ**: boolean
- **Neustart erforderlich**: nein
- **System**: nein
- **Optional**: ja
- **Beispielwert**: true
- **Seit**: 6.4.0

mailbox.default.session.mail.imap.timeout

- **Modul**: cmas-nimh
- **Beschreibung**: Beispiel für javax.mail-Property
- **Typ**: integer
- **Neustart erforderlich**: nein

- **System**: nein
- **Optional**: ja
- **Beispielwert**: 120
- **Seit**: 6.4.0

mailbox.default.session.mail.mime.address.strict

- **Modul**: cmas-nimh
- **Beschreibung**: Beispiel für javax.mail-Property - Gegenstück zum alten *mail.mime.strict* von Mule, erlaubt das nicht so strikte Parsen des E-Mail-Headers.
- **Typ**: boolean
- **Neustart erforderlich**: nein
- **System**: nein
- **Optional**: ja
- **Beispielwert**: true
- **Seit**: 6.4.0

mailbox.default.session.mail.pop3.timeout

- **Modul**: cmas-nimh
- **Beschreibung**: Beispiel für javax.mail-Property
- **Typ**:
- **Neustart erforderlich**:
- **System**:
- **Optional**:
- **Beispielwert**:
- **Seit**: 6.4.0

mailbox.default.session.mail.<protocol>.partialfetch

- **Modul**: cmas-nimh
- **Beschreibung**: z. B. mailbox.default.session.mail.imaps.partialfetch
- **Typ**: boolean
- **Neustart erforderlich**: nein
- **System**: nein
- **Optional**: ja
- **Beispielwert**: false
- **Seit**:

mailbox.default.task.delete.read.messages

- **Modul**: cmas-nimh
- **Beschreibung**: Legt fest, ob Nachrichten nach dem Lesen aus dem Postfach entfernt werden sollen. Beim IMAP-Protokoll werden Nachrichten standardmäßig als SEEN gekennzeichnet. Bei POP3 wird die Nachricht nur gelöscht, wenn der Wert auf true gesetzt ist. Anderenfalls bleibt die Nachricht auf dem Server, was zu einer Endlos-Leseschleife führt. Standardwert: false.
- **Typ**: boolean
- **Neustart erforderlich**: nein
- **System**: nein
- **Optional**: ja
- **Beispielwert**: false
- **Seit**: 6.4.0

mailbox.default.task.enabled

- **Modul**: cmas-nimh
- **Beschreibung**: Mit dieser System-Property kann der Service Thread eines bestimmten Pollers deaktiviert werden. Standardwert: true.
- **Typ**: boolean
- **Neustart erforderlich**: nein
- **System**: nein
- **Optional**: ja
- **Beispielwert**: false
- **Seit**: 6.4.0

mailbox.default.task.interval.seconds

- **Modul**: cmas-nimh
- **Beschreibung**: Standard-Intervall für den Abruf von Postfächern. Standardwert: 60 Sekunden
- **Typ**: integer
- **Neustart erforderlich**: nein
- **System**: nein
- **Optional**: ja
- **Beispielwert**: 60
- **Seit**: 6.4.0

mailbox.default.task.max.message.size

- **Modul**: cmas-nimh
- **Beschreibung**: Maximale Größe von E-Mails (d. h. E-Mail plus Attachments). E-Mails, die größer als dieser Wert sind, werden nicht automatisch von NIMH verarbeitet. Sie werden in der Datenbank (Tabelle *cmas_nimh_archived_mail*) gespeichert und erscheinen daher in den E-Mail-Backups im Admin Tool (siehe Abschnitt *E-Mail-Backups* im *ConSol CM Administratorhandbuch*). Von dort können sie erneut gesendet, in das Dateisystem geladen oder gelöscht werden. Für diese Operationen ist die Größe der Nachricht nicht relevant. Standardmäßig auf 10 MB (10485760) gesetzt.
- **Typ**: integer
- **Neustart erforderlich**: nein
- **System**: nein
- **Optional**: ja
- **Beispielwert**: 10485760
- **Seit**: 6.4.0

mailbox.default.task.max.messages.per.run

- **Modul**: cmas-nimh
- **Beschreibung**: Anzahl der Nachrichten, die gleichzeitig aus dem Postfach abgeholt werden. Muss mit dem Transaktions-Timeout korrelieren. Standardwert: 20.
- **Typ**: integer
- **Neustart erforderlich**: nein
- **System**: nein
- **Optional**: ja
- **Beispielwert**: 60
- **Seit**: 6.4.0

mailbox.default.task.timeout.seconds

- **Modul**: cmas-nimh
- **Beschreibung**: Nach dieser Zeit (der Inaktivität) wird der Service Thread als beschädigt betrachtet und automatisch neu gestartet. Standardwert: 120 Sekunden
- **Typ**: integer
- **Neustart erforderlich**: nein
- **System**: nein
- **Optional**: ja
- **Beispielwert**: 60
- **Seit**: 6.4.0

mailbox.default.task.transaction.timeout.seconds

- **Modul**: cmas-nimh
- **Beschreibung**: Standard-Transaktions-Timeout für Transaktionen, die E-Mails abrufen. Sollte mit der Anzahl der Nachrichten, die gleichzeitig abgeholt werden, korrelieren. Standardwert: 60 Sekunden
- **Typ**: integer
- **Neustart erforderlich**: nein
- **System**: nein
- **Optional**: ja
- **Beispielwert**: 60
- **Seit**: 6.4.0

mailbox.polling.threads.mail.log.enabled

- **Modul**: cmas-nimh
- **Description**: Ermöglicht das Protokollieren von E-Mails, was in Cluster-Umgebungen besonders wichtig ist (wird dort als Semaphor genutzt).
- **Typ**: boolean
- **Neustart erforderlich**: nein
- **System**: nein
- **Optional**: ja
- **Beispielwert**: true (Standardwert)
- **Seit**: 6.9.4.1

mailbox.polling.threads.number

- **Modul**: cmas-nimh
- **Beschreibung**: Anzahl der Threads für den Zugriff auf Postfächer. Standardwert: 1.
- **Typ**: integer
- **Neustart erforderlich**: nein
- **System**: nein
- **Optional**: ja
- **Beispielwert**: 1
- **Seit**: 6.4.0

mailTemplateAboveQuotedText

- **Modul**: cmweb-server-adapter
- **Beschreibung**: Legt das Verhalten der E-Mail-Vorlagen im Ticket-E-Mail-Editor fest, wenn eine andere E-Mail zitiert wird, d. h. auf diese geantwortet oder diese weitergeleitet wird. Wird oft verwendet, um die Signatur korrekt zu platzieren.
- **Typ**: boolean
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: false
- **Seit**: 6.2.4

max.licences.perUser

- **Modul**: cmas-core-server
- **Beschreibung**: Setzt die maximale Anzahl an Lizenzen, die ein einzelner Benutzer benutzen kann (z. B. durch Anmelden in einem anderen Browser). Standardmäßig ist dieser Wert nicht beschränkt.
- **Typ**: integer
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: ja
- **Beispielwert**: 10
- **Seit**: 6.8.4.5

maxSizePerPagemapInMegaBytes

- **Modul**: cmweb-server-adapter
- **Beschreibung**: Maximale Größe (in MB) für jede Wicket Pagemap.
- **Typ**: integer
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: 15
- **Seit**: 6.3.5

monitoring.engineer.login

- **Modul**: cmas-core-server
- **Beschreibung**: Login des Monitoring-Bearbeiters.

- **Typ**: string
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: ja
- **Beispielwert**: nagios
- **Seit**: 6.9.3.0

monitoring.unit.login

- **Modul**: cmas-core-server
- **Beschreibung**: Login der Monitoring-Unit.
- **Typ**: string
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: ja
- **Beispielwert**: nagios
- **Seit**: 6.9.3.0

nimh.enabled

- **Modul**: cmas-core-server
- **Beschreibung**: Aktiviert den NIMH-Dienst. Im Cluster muss die Node-ID angehängt werden, z. B. *nimh.enabled.NODEID = true*.
- **Typ**: boolean
- **Neustart erforderlich**: nein
- **System**: nein
- **Optional**: ja
- **Beispielwert**: false
- **Seit**: 6.9.4.0

notification.error.description

- **Modul**: cmas-dwh-server
- **Beschreibung**: Text für E-Mails mit Fehlermeldungen des DWH.
- **Typ**: string
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein

- **Beispielwert**: Fehler aufgetreten
- **Seit**: 6.0.1

notification.error.from

- **Modul**: cmas-dwh-server
- **Beschreibung**: From-Adresse für E-Mails mit Fehlermeldungen des DWH.
- **Typ**: string
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: ja
- **Seit**: 6.0.1

notification.error.subject

- **Modul**: cmas-dwh-server
- **Beschreibung**: Betreff für E-Mails mit Fehlermeldungen des DWH.
- **Typ**: string
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: Fehler aufgetreten
- **Seit**: 6.0.1

notification.error.to

- **Modul**: cmas-dwh-server
- **Beschreibung**: To-Adresse für E-Mails mit Fehlermeldungen des DWH.
- **Typ**: string
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: myuser@consol.de
- **Seit**: 6.0.1

notification.finished_successfully.description

- **Modul**: cmas-dwh-server
- **Beschreibung**: Text für E-Mails des DWHs, wenn eine Übertragung erfolgreich beendet wurde.
- **Typ**: string
- **Neustart erforderlich**: nein

- **System**: ja
- **Optional**: nein
- **Beispielwert**: Übertragung erfolgreich beendet
- **Seit**: 6.0.1

notification.finished_successfully.from

- **Modul**: cmas-dwh-server
- **Beschreibung**: From-Adresse für E-Mails des DWHs, wenn eine Übertragung erfolgreich beendet wurde.
- **Typ**: string
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: ja
- **Seit**: 6.0.1

notification.finished_successfully.subject

- **Modul**: cmas-dwh-server
- **Beschreibung**: Betreff für E-Mails des DWHs, wenn eine Übertragung erfolgreich beendet wurde.
- **Typ**: string
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: Übertragung erfolgreich beendet
- **Seit**: 6.0.1

notification.finished_successfully.to

- **Modul**: cmas-dwh-server
- **Beschreibung**: To-Adresse für E-Mails des DWHs, wenn eine Übertragung erfolgreich beendet wurde.
- **Typ**: string
- **Neustart erforderlich**: ja
- **System**: ja
- **Optional**: nein
- **Beispielwert**: myuser@consol.de
- **Seit**: 6.0.1

notification.finished_unsuccessfully.description

- **Modul**: cmas-dwh-server
- **Beschreibung**: Text für E-Mails des DWHs, wenn eine Übertragung nicht erfolgreich beendet wurde.
- **Typ**: string
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: Übertragung nicht erfolgreich beendet
- **Seit**: 6.0.1

notification.finished_unsuccessfully.from

- **Modul**: cmas-dwh-server
- **Beschreibung**: From-Adresse für E-Mails des DWHs, wenn eine Übertragung nicht erfolgreich beendet wurde.
- **Typ**: string
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: ja
- **Seit**: 6.0.1

notification.finished_unsuccessfully.subject

- **Modul**: cmas-dwh-server
- **Beschreibung**: Betreff für E-Mails des DWHs, wenn eine Übertragung nicht erfolgreich beendet wurde.
- **Typ**: string
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: Übertragung nicht erfolgreich beendet
- **Seit**: 6.0.1

notification.finished_unsuccessfully.to

- **Modul**: cmas-dwh-server
- **Beschreibung**: To-Adresse für E-Mails des DWHs, wenn eine Übertragung nicht erfolgreich beendet wurde.
- **Typ**: string

- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: myuser@consol.de
- **Seit**: 6.0.1

notification.host

- **Modul**: cmas-dwh-server
- **Description**: Hostname des E-Mail-Servers (SMTP) für das Senden von DWH-E-Mails.
- **Typ**: string
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: ja
- **Beispielwert**: myserver.consol.de
- **Seit**: 6.0.1

notification.password

- **Modul**: cmas-dwh-server
- **Beschreibung**: Passwort für das Senden von DWH-E-Mails. (optional)
- **Typ**: string
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: ja
- **Seit**: 6.0.1

notification.port

- **Modul**: cmas-dwh-server
- **Beschreibung**: SMTP-Port für das Senden von DWH-E-Mails.
- **Typ**: string
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: ja
- **Beispielwert**: 25
- **Seit**: 6.0.1

notification.protocol

- **Modul**: cmas-dwh-server
- **Beschreibung**: Das Protokoll, das für das Senden von E-Mails aus dem DWH verwendet wird.
- **Typ**: string
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: ja
- **Beispielwert**: pop3\

notification.username

- **Modul**: cmas-dwh-server
- **Beschreibung**: Benutzername (SMTP) für das Senden von DWH-E-Mails.
- **Typ**: string
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: ja
- **Beispielwert**: myuser
- **Seit**: 6.0.1

outdated.lock.age

- **Modul**: cmas-workflow-jbpm
- **Beschreibung**:
- **Typ**: integer
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: 60000
- **Entfernt in**: 6.8.0
- **Ersetzt durch**: cmas-workflow-engine, jobExecutor.lockTimeout.seconds

pagemapLockDurationInSeconds

- **Modul**: cmweb-server-adapter
- **Beschreibung**: Anzahl der Sekunden, die vergehen müssen, bevor eine Pagemap als zu lange gelockt angesehen wird.
- **Typ**: integer
- **Neustart erforderlich**: ja

- **System**: ja
- **Optional**: ja
- **Beispielwert**: 60
- **Seit**: 6.7.3

policy.password.age

- **Modul**: cmas-core-security
- **Beschreibung**: Definiert (in Tagen), wie alt das Passwort sein darf.
- **Typ**: integer
- **Neustart erforderlich**: nein
- **System**: nein
- **Optional**: ja
- **Beispielwert**: 5500 (15 Jahre, Standardwert)
- **Seit**: 6.10.1.0

policy.password.pattern

- **Modul**: cmas-core-security
- **Beschreibung**: Definiert das Passwortmuster.
- **Typ**: string
- **Neustart erforderlich**: nein
- **System**: nein
- **Optional**: ja
- **Beispielwert**: "^.3,$" (Standardwert)
- **Seit**: 6.10.1.0

policy.rotation.ratio

- **Modul**: cmas-core-security
- **Beschreibung**: Legt fest, wie oft sich das Passwort wiederholen darf. Ist der Wert z. B. 10, darf das neue Passwort nicht unter den zehn letzten Passwörtern des Benutzers sein.
- **Typ**: integer
- **Neustart erforderlich**: nein
- **System**: nein
- **Optional**: ja
- **Beispielwert**: 1 (Standardwert)
- **Seit**: 6.10.1.0

policy.username.case.sensitive

- **Modul**: cmas-core-security
- **Beschreibung**: Legt fest, ob bei Benutzernamen die Groß- und Kleinschreibung beachtet werden muss.
- **Typ**: boolean
- **Neustart erforderlich**: nein
- **System**: nein
- **Optional**: ja
- **Beispielwert**: true (Standardwert)
- **Seit**: 6.10.1.0

postActivityExecutionScriptName

- **Modul**: cmweb-server-adapter
- **Beschreibung**: Definiert den Namen des Skripts, das nach jeder Workflow-Aktivität ausgeführt wird (siehe Abschnitt *PostActivityExecutionScript* im *ConSol CM Administratorhandbuch*. Wenn kein Skript ausgeführt werden soll, lassen Sie diesen Wert leer.
- **Typ**: string
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: postActivityExecutionHandler
- **Seit**: 6.2.0

queue.polling.threads.number

- **Modul**: cmas-nimh
- **Beschreibung**: Anzahl der Threads, die zur Überwachung der E-Mail-Warteschlange gestartet werden. Standardwert: 1.
- **Typ**: integer
- **Neustart erforderlich**: nein
- **System**: nein
- **Optional**: ja
- **Beispielwert**: 1
- **Seit**: 6.4.0

queue.polling.threads.shutdown.timeout.seconds

- **Modul**: cmas-nimh
- **Beschreibung**: Wartezeit nach dem Shutdown-Signal. Wenn der Timeout erreicht ist, wird der Thread beendet. Standardwert: 60.
- **Typ**: integer
- **Neustart erforderlich**: nein
- **System**: nein
- **Optional**: ja
- **Beispielwert**: 60
- **Seit**: 6.4.0

queue.polling.threads.watchdog.interval.seconds

- **Modul**: cmas-nimh
- **Beschreibung**: Intervall des Watchdog Threads. Standardwert: 30.
- **Typ**: integer
- **Neustart erforderlich**: nein
- **System**: nein
- **Optional**: ja
- **Beispielwert**: 30
- **Seit**: 6.4.0

queue.task.error.pause.seconds

- **Modul**: cmas-nimh
- **Beschreibung**: Anzahl der Sekunden, die der Queue-Poller nach einem Infrastrukturfehler (z. B. der Datenbank) wartet. Standardwert: 180 Sekunden
- **Typ**: integer
- **Neustart erforderlich**: nein
- **System**: nein
- **Optional**: ja
- **Beispielwert**: 180
- **Seit**: 6.4.0

queue.task.interval.seconds

- **Modul**: cmas-nimh
- **Beschreibung**: Thread-Intervall zur Überwachung der Haupt-E-Mail-Warteschlange. Standardwert: 15.
- **Typ**: integer

- **Neustart erforderlich**: nein
- **System**: nein
- **Optional**: ja
- **Beispielwert**: 15
- **Seit**: 6.4.0

queue.task.max.retries

- **Modul**: cmas-nimh
- **Beschreibung**: Maximale Anzahl der Versuche, eine E-Mail nach einer Exception erneut zu verarbeiten. Ist diese erreicht, wird die E-Mail archiviert. Die archivierte E-Mail kann über das NIMH API (oder das Admin Tool) wieder aktiviert werden.
- **Typ**: integer
- **Neustart erforderlich**: nein
- **System**: nein
- **Optional**: ja
- **Beispielwert**: 10
- **Seit**: 6.4.0

queue.task.timeout.seconds

- **Modul**: cmas-nimh
- **Beschreibung**: Nach dieser Zeit (der Inaktivität) wird der Service Thread als beschädigt betrachtet und automatisch neu gestartet. Standardwert: 600 Sekunden
- **Typ**: integer
- **Neustart erforderlich**: nein
- **System**: nein
- **Optional**: ja
- **Beispielwert**: 600
- **Seit**: 6.4.0

queue.task.transaction.timeout.seconds

- **Modul**: cmas-nimh
- **Beschreibung**: Transaktions-Timeout für die E-Mail-Verarbeitung in der Warteschlange. Standardwert: 60.
- **Typ**: integer
- **Neustart erforderlich**: nein
- **System**: nein
- **Optional**: ja

- **Beispielwert**: 60
- **Seit**: 6.4.0

queuesExcludedFromGS

- **Modul**: cmweb-server-adapter
- **Beschreibung**: Durch Kommas getrennte Liste von Queue-Namen, die von der Schnellsuche ausgeschlossen werden sollen.
- **Typ**: string
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: ja
- **Seit**: 6.0

refreshTimeInCaseOfConcurrentRememberMeRequests

- **Modul**: cmas-workflow-jbpm
- **Beschreibung**: Legt die Aktualisierungszeit (in Sekunden) fest, nach der die Seite im Falle von gleichzeitigen Anfragen von Angemeldet bleiben neu geladen wird. Diese Funktion verhindert, dass ein Benutzer zu viele Lizenzen in Anspruch nimmt. Erhöhen Sie die Zeit, wenn immer noch Sessions belegt werden.
- **Typ**: integer
- **Neustart erforderlich**: ja
- **System**: ja
- **Optional**: ja
- **Beispielwert**: 5
- **Seit**: 6.8.2

rememberMeLifetimeInMinutes

- **Modul**: cmweb-server-adapter
- **Beschreibung**: Lebensdauer für *Angemeldet bleiben* in Minuten.
- **Typ**: integer
- **Neustart erforderlich**: ja
- **System**: ja
- **Optional**: nein
- **Beispielwert**: 1440
- **Seit**: 6.0

request.scope.transaction

- **Modul**: cmweb-server-adapter
- **Beschreibung**: Ermöglicht die Deaktivierung von Scope-Transaktionen für Requests. Standardmäßig wird pro Request eine Transaktion verwendet. Wenn Sie diese Property auf *false* setzen, wird eine Transaktion pro Aufruf einer Service-Methode erzeugt.
- **Typ**: boolean
- **Neustart erforderlich**: ja
- **System**: ja
- **Optional**: ja
- **Beispielwert**: true
- **Seit**: 6.8.1

resetCode.expiriationPeriod

- **Modul**: cmas-core-security
- **Beschreibung**: Definiert die Gültigkeitsdauer des Links zum Zurücksetzen des Passworts in CM.Track.
- **Typ**: Integer
- **Neustart erforderlich**: nein
- **System**: nein
- **Optional**: ja
- **Beispielwert**: 86400000 (Standardwert, 24 Stunden)
- **Seit**: 6.10.1

resource.replace.batchSize

- **Modul**: cmas-core-server
- **Beschreibung**: Legt die Anzahl der Objekte fest, die in einer Ressourcen-Ersetzen-Aktion verarbeitet werden.
- **Typ**: integer
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: 5
- **Seit**: 6.10.0.0

resource.replace.timeout

- **Modul**: cmas-core-server
- **Beschreibung**: Transaktions-Timeout (in Sekunden) für einen Schritt einer Ressourcen-Ersetzen-Aktion.
- **Typ**: integer
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: 120
- **Seit**: 6.10.0.0

scene

- **Modul**: cmas-setup-scene
- **Beschreibung**: Szenariodatei, die während des Setups importiert wurde (kann leer gelassen werden).
- **Typ**: string
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: vfszip:/P:/dist/target/jboss/server/cmas/deploy/cm-dist-6.5.1-SNAPSHOT.ear/APP-INF/lib/dist-scene-6.5.1-SNAPSHOT.jar/META-INF/cmas/scenes/helpdesk-sales_scene.jar/
- **Seit**: 6.0

script.logging.threshold.seconds

- **Modul**: cmas-core-server
- **Beschreibung**: Legt die Zeit (in Sekunden) fest, nach deren Ablauf während einer Skript-Ausführung eine Warnung in der Log-Datei generiert wird.
- **Typ**: integer
- **Neustart erforderlich**: nein
- **System**: nein
- **Optional**: ja
- **Beispielwert**: 10 (Standardwert)
- **Seit**: 6.10.1.0

searchPageSize

- **Modul**: cmweb-server-adapter
- **Beschreibung**: Standardgröße der Seiten für Suchergebnisse.
- **Typ**: integer
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: 20
- **Seit**: 6.0

searchPageSizeOptions

- **Modul**: cmweb-server-adapter
- **Beschreibung**: Optionen für die Größe der Seiten für Suchergebnisse.
- **Typ**: string
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: 10|20|30|40|50|75|100
- **Seit**: 6.0

security.fields.customer.exposure.check.enabled

- **Modul**: cmas-restapi-core
- **Beschreibung**: Aktiviert die Prüfung der Annotation *customer exposure* für Benutzerdefinierte Felder.
- **Typ**: boolean
- **Neustart erforderlich**: nein
- **System**: nein
- **Optional**: ja
- **Beispielwert**: true (Standardwert)
- **Seit**: 6.10.5.4

server.session.archive.reaper.interval

- **Modul**: cmas-core-server
- **Beschreibung**: Reaper-Intervall (in Sekunden) von archivierten Server-Sessions.
- **Typ**: integer
- **Neustart erforderlich**: nein

- **System**: ja
- **Optional**: ja
- **Beispielwert**: 60
- **Seit**: 6.7.1

server.session.archive.timeout

- **Modul**: cmas-core-server
- **Beschreibung**: Timeout der Gültigkeit der Server-Session-Archive (in Tagen). Nach diesem Zeitraum werden die Informationen zur Session aus der Datenbank entfernt.
- **Typ**: integer
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: 31
- **Seit**: 6.7.1

server.session.reaper.interval

- **Modul**: cmas-core-server
- **Beschreibung**: Interval (in Sekunden), in dem der sog. Reaper inaktive (= beendete) Server-Sessions löscht.
- **Typ**: integer
- **Neustart erforderlich**: Nur Session Service
- **System**: ja
- **Optional**: nein
- **Beispielwert**: 60
- **Seit**: 6.6.1, 6.7.1

server.session.timeout

- **Modul**: cmas-core-server
- **Beschreibung**: Server-Session-Timeout (in Sekunden) für verbundene Clients. Jeder Client kann dieses Timeout mit benutzerdefinierten Werten mittels seiner ID (ADMIN_TOOL, WEB_CLIENT, WORKFLOW_EDITOR, TRACK (vor 6.8 bitte PORTER verwenden), ETL, REST) überschreiben. Diese wird an den Namen der System-Property angehängt, z. B. *server.session.timeout.ADMIN_TOOL*.
  Siehe auch die Attribute der Seitenanpassung *updateTimeServerSessionActivityEnabled* und *updateTimeServerSessionActivity*, beide vom Typ *cmApplicationCustomization*.
- **Typ**: integer
- **Neustart erforderlich**: nein

- **System**: ja
- **Optional**: nein
- **Beispielwert**: 1800
- **Seit**: 6.6.1, 6.7.1

Detaillierte Erklärung für das Admin Tool:

- server.session.timeout.ADMIN_TOOL
  Definiert den Zeitraum, den der Server eine Session im Admin Tool als gültig betrachtet, wenn keine Aktivität im Admin Tool der Session erfolgt. Das Admin Tool kennt diesen Wert nicht, es bemerkt nur eine ungültige Session, wenn es länger keine Aktivität gegeben hat.
- admin.tool.session.check.interval
  Definiert den Zeitraum zwischen zwei Überprüfungen durch das Admin Tool, ob der Server die Session noch als gültig betrachtet.

Wenn zum Beispiel admin.tool.session.check.interval = 60, fragt das Admin Tool den Server einmal pro Minute, ob die Session noch aktiv/gültig ist. Wenn server.session.timeout.ADMIN_TOOL = 600 erhält das Admin Tool die Antwort, dass die Session ungültig ist, nach zehn Minuten der Inaktivität.

serverPoolingInterval

- **Modul**: cmweb-server-adapter
- **Beschreibung**: Definiert die Zeit in Sekunden, nach der der Pooling-Server die Caches auf dem Web-Layer ungültig macht.
- **Typ**: integer
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: 5
- **Seit**: 6.1.0

skip-ticket

- **Modul**: cmas-dwh-server
- **Beschreibung**: Tickets werden bei der Übertragung/Aktualisierung nicht übermittelt.
- **Typ**: boolean
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: false
- **Seit**: 6.6.19
- **Entfernt in**: 6.8.1

skip-ticket-history

- **Modul**: cmas-dwh-server
- **Beschreibung**: Ticketprotokoll wird bei der Übertragung/Aktualisierung nicht übermittelt.
- **Typ**: boolean
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: false
- **Seit**: 6.6.19
- **Entfernt in**: 6.8.1

skip-unit

- **Modul**: cmas-dwh-server
- **Beschreibung**: Units werden bei der Übertragung/Aktualisierung nicht übermittelt.
- **Typ**: boolean
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: false
- **Seit**: 6.6.19
- **Entfernt in**: 6.8.1

skip-unit-history

- **Modul**: cmas-dwh-server
- **Beschreibung**: Unit-Protokoll wird bei der Übertragung/Aktualisierung nicht übermittelt.
- **Typ**: boolean
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: false
- **Seit**: 6.6.19
- **Entfernt in**: 6.8.1

skip.wfl.transfer.cleanup

- **Modul**: cmas-core-server
- **Beschreibung**: Wenn diese System-Property auf *true* gesetzt wird, wird das Workflow-Cleanup nach der Übertragung übersprungen.
- **Typ**: boolean
- **Neustart erforderlich**: nein
- **System**: nein
- **Optional**: ja
- **Beispielwert**: false (Standardwert)
- **Seit**: 6.9.4.1

split.history

- **Modul**: cmas-dwh-server
- **Beschreibung**: Ändert das SQL-Statement dahingehend, dass Ticketprotokolle während der Übertragung an das DWH nicht für alle Tickets auf einmal abgeholt werden, sondern für ein Ticket pro SQL-Statement.
- **Typ**: boolean
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: ja
- **Beispielwert**: false
- **Seit**: 6.8.0

start.groovy.task.enabled

- **Modul**: cmas-app-admin-tool
- **Beschreibung**: Um Skripte vom Typ *Task* im Admin Tool (Navigationsgruppe *Dienste*, Navigationselement *Task-Ausführung*) ausführen zu können, ist es nötig, den Button *Start* zu aktivieren, der standardmäßig ausgeblendet ist. Setzen Sie hierfür diese System-Property auf *true*.
- **Typ**: boolean
- **Neustart erforderlich**: nein
- **System**: nein
- **Optional**: ja
- **Beispielwert**: true
- **Seit**: 6.9.4.0

supportEmail

- **Modul**: cmweb-server-adapter
- **Beschreibung**:
- **Typ**: string
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: ja
- **Seit**: 6.0
- **Entfernt in**: 6.11.0.1

synchronize.master.address

- **Modul**: cmas-core-index-common
- **Beschreibung**: Wert der Java-System-Property *-Dcmas.http.host.port*, die angibt, unter welcher URL der Index-Master-Server erreichbar ist. Standard ist Null. Seit CM-Version 6.6.17 ist dieser Wert beim Setup konfigurierbar, um den initialen Index-Master-Server zu bestimmen. Bitte beachten Sie, dass das Verändern dieses Wertes nur erlaubt ist, wenn alle Cluster-Nodes zum Empfang von Index-Veränderungen gestoppt sind.
- **Typ**: integer
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: ja
- **Beispielwert**: 127.0.0.1:80
- **Seit**: 6.6.0

synchronize.master.security.token

- **Modul**: cmas-core-index-common
- **Beschreibung**: Das Passwort für den URL-Zugriff auf den Index-Snapshot, z. B. für die Index-Synchronisierung oder für Backups.
- **Typ**: string
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: ja
- **Beispielwert**: token
- **Seit**: 6.6.0

synchronize.master.security.user

- **Modul**: cmas-core-index-common
- **Beschreibung**: Der Benutzername für den URL-Zugriff auf den Index-Snapshot, z. B. für die Index-Synchronisierung oder für Backups.
- **Typ**: string
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: ja
- **Beispielwert**: user
- **Seit**: 6.6.0

synchronize.master.timeout.minutes

- **Modul**: cmas-core-index-common
- **Beschreibung**: Legt fest, wie oft die Index-Synchronisierung ausgehend vom aktuellen Master-Server fehlschlagen darf, bis ein neuer Master für die Index-Reparatur ausgewählt wird. Standardwert ist 5. Seit CM-Version 6.6.17 ist dieser Wert im Setup konfigurierbar, wobei 0 bedeutet, dass der Master-Server nie geändert wird (Failover-Mechanismus deaktiviert).
- **Typ**: integer
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: 5
- **Seit**: 6.6.0

synchronize.megabits.per.second

- **Modul**: cmas-core-index-common
- **Beschreibung**: Legt fest, wie viel Bandbreite der Master-Server verbrauchen darf, um Index-Änderungen an die Slave-Server zu übermitteln. Standardwert ist 85. Nutzen Sie nicht die gesamte verfügbare Bandbreite, um die Index-Änderungen zwischen den Hosts zu übermitteln, da dies dafür sorgen kann, dass die Nodes des Clusters nicht mehr synchron sind.
- **Typ**: integer
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: 85
- **Seit**: 6.6.0

synchronize.sleep.millis

- **Modul**: cmas-core-index-common
- **Beschreibung**: Legt fest, wie oft jeder Slave-Server den Master-Server auf Änderungen am Index abfragt. Standardwert ist 1000.
- **Typ**: integer
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: 1000
- **Seit**: 6.6.0

task.panel.refresh.interval.seconds

- **Modul**: cmas-app-admin-tool
- **Beschreibung**: Zeit in Sekunden, nachdem die Task-Liste (im Admin Tool) des Task Execution Framework aktualisiert wird.
- **Typ**: Integer
- **Neustart erforderlich**: nein
- **System**: nein
- **Optional**: nein
- **Beispielwert**: 10
- **Seit**: 6.10.5.3 (wird beim Update von einer Version vor 6.10.5.3 nicht automatisch hinzugefügt!)

themeOverlay

- **Modul**: cmweb-server-adapter
- **Beschreibung**: Name des verwendeten Themen-Overlays.
- **Typ**: string
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: ja
- **Beispielwert**: consolINT
- **Seit**: 6.0

ticket.delete.timeout

- **Modul**: cmas-core-server
- **Beschreibung**: Transaktions-Timeout (in Sekunden) beim Löschen von Tickets.
- **Typ**: integer
- **Neustart erforderlich**: nein

- **System**: ja
- **Optional**: nein
- **Beispielwert**: 60
- **Seit**: 6.1.3

ticketListRefreshIntervalInSeconds

- **Modul**: cmweb-server-adapter
- **Beschreibung**: Aktualisierungsintervall für die Ticketliste (in Sekunden).
- **Typ**: integer
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: 180
- **Seit**: 6.0

ticketListSizeLimit

- **Modul**: cmweb-server-adapter
- **Beschreibung**: Maximale Anzahl von Tickets in der Ticketliste.
- **Typ**: integer
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: 100
- **Seit**: 6.0

tickets.delete.size

- **Modul**: cmas-core-server
- **Beschreibung**: Definiert die Anzahl der Tickets, die pro Transaktion gelöscht werden. Standardmäßig ist dieser Wert 10.
- **Typ**: integer
- **Neustart erforderlich**: Nur Session Service
- **System**: ja
- **Optional**: nein
- **Beispielwert**: 10
- **Seit**: 6.8.1

transaction.timeout.minutes

- **Modul**: cmas-core-server
- **Beschreibung**: Setzt den Timeout für den Task Execution Service des TEF, d. h. ein Durchlauf eines Tasks muss vor dem Ablauf dieser Zeitspanne abgeschlossen sein. Die Änderungen sind nur für neue Aufgaben sichtbar, deren Ausführung nach der Konfigurationsänderung beginnt.
- **Typ**: integer
- **Neustart erforderlich**: nein
- **System**: nein
- **Optional**: ja
- **Beispielwert**: 10*3600 (10 Stunden - Standardwert)
- **Seit**: 6.10

unit.replace.batchSize

- **Modul**: cmas-core-server
- **Beschreibung**: Legt die Anzahl der Objekte fest, die in einer Unit-Ersetzen-Aktion verarbeitet werden.
- **Typ**: integer
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: 5
- **Seit**: 6.8.2

unit.replace.timeout

- **Modul**: cmas-core-server
- **Beschreibung**: Transaktions-Timeout (in Sekunden) für einen Schritt einer Unit-Ersetzen-Aktion.
- **Typ**: integer
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: 120
- **Seit**: 6.8.2

unit.transfer.order

- **Modul**: cmas-dwh-server
- **Beschreibung**: Legt fest, in welcher Reihenfolge Datenobjektgruppen an das DWH übertragen werden.

- **Typ**: string
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: ja
- **Beispielwert**: company;customer
- **Seit**: 6.6.19
- **Entfernt in**: 6.8.1

unitIndexSearchResultSizeLimit

- **Modul**: cmweb-server-adapter
- **Beschreibung**: Maximale Anzahl der Units in der Ergebnisliste, wenn nach Units gesucht wird (Beispiel: Kontaktsuche).
- **Typ**: integer
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: 5
- **Seit**: 6.0

unused.content.remover.cluster.node.id

- **Modul**: cmas-core-server
- **Beschreibung**: Wert eines *cmas.clusternode.id*, der angibt, welcher Node nicht verwendete Ticket-Attachments und Unit-Inhaltseinträge entfernt.
- **Typ**: string
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: ja
- **Beispielwert**: 1 (in der Annahme, dass der Cluster-Node mit dem Parameter *-Dcmas.clusternode.id=1* gestartet ist)
- **Seit**: 6.9.0.0

unused.content.remover.enabled

- **Modul**: cmas-core-server
- **Beschreibung**: Legt fest, ob das Entfernen ungenutzter Ticket-Attachments und Unit-Inhaltseinträge durchgeführt werden soll.
- **Typ**: boolean
- **Neustart erforderlich**: nein

- **System**: ja
- **Optional**: nein
- **Beispielwert**: true
- **Seit**: 6.9.0.0

unused.content.remover.polling.minutes

- **Modul**: cmas-core-server
- **Beschreibung**: Legt fest, wie oft überprüft werden soll, ob ungenutzte Ticket-Attachments und Unit-Inhaltseinträge zum Entfernen vorhanden sind.
- **Typ**: integer
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: 15
- **Seit**: 6.9.0.0

unused.content.remover.ttl.minutes

- **Modul**: cmas-core-server
- **Beschreibung**: Mindestzeitraum, nach dem ungenutzte Ticket-Attachments und Unit-Inhaltseinträge entfernt werden können.
- **Typ**: integer
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: 1440
- **Seit**: 6.9.0.0

urlLogoutPath

- **Modul**: cmweb-server-adapter
- **Beschreibung**: URL die verwendet wird, wenn sich der Bearbeiter abmeldet. (wenn kein Wert gesetzt wird, wird nach dem Abmelden die Anmeldeseite angezeigt.)
- **Typ**: string
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: ja
- **Beispielwert**: http://intranet.consol.de
- **Seit**: 6.3.1

warmup.executor.enabled

- **Modul**: cmas-core-server
- **Beschreibung**: Legt fest, ob der Server beim Start asynchron anlaufen soll und nebenher andere Aufgaben erledigt (z. B. interne Caches füllen).
- **Typ**: boolean
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: true
- **Seit**: 6.9.4.2

webSessionTimeoutInMinutes

- **Modul**: cmweb-server-adapter
- **Beschreibung**: Session-Timeout in Minuten.
- **Typ**: integer
- **Neustart erforderlich**: ja
- **System**: ja
- **Optional**: nein
- **Beispielwert**: 180
- **Entfernt in**: 6.7.1
- **Ersetzt durch**: cmas-core-server, server.session.timeout

wicketAjaxRequestHeaderFilterEnabled

- **Modul**: cmweb-server-adapter
- **Beschreibung**: Aktiviert Filter für Wicket AJAX-Anfragen, die auf veralteten Seiten mit Wicket 1.4-Skripten (CM-Version vor 6.8.0) stammen, nach einer Aktualisierung auf CM-Versionen ab 6.8.0.
- **Typ**: boolean
- **Neustart erforderlich**: ja
- **System**: ja
- **Optional**: ja
- **Beispielwert**: false
- **Seit**: 6.8.1

G.2.2 Liste der System-Properties nach Modul

In diesem Kapitel sind die System-Properties der folgenden Module aufgeführt:

G.2.2.1 cmas-app-admin-tool (Modul)

admin.tool.session.check.interval

- **Modul**: cmas-app-admin-tool
- **Beschreibung**: Intervall, in dem inaktive (beendete) Sitzungen im Admin Tool überprüft werden (in Sekunden).
- **Typ**: integer
- **Neustart erforderlich**: ja
- **System**: ja
- **Optional**: nein
- **Beispielwert**: 30
- **Seit**: 6.7.5

autocomplete.enabled

- **Modul**: cmas-app-admin-tool
- **Beschreibung**: Wenn diese System-Property fehlt oder der Wert *false* ist, wird das Navigationselement *Adress-Vervollständigung* im Admin Tool ausgeblendet.
- **Typ**: boolean
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: ja
- **Beispielwert**: true
- **Seit**: 6.9.2.0

delete.ticket.enabled

- **Modul**: cmas-app-admin-tool
- **Beschreibung**: Steuert, ob in der Ticketverwaltung im Admin Tool der Menüpunkt *Tickets löschen* im Kontextmenü der Ticketliste angezeigt wird.
- **Typ**: boolean
- **Neustart erforderlich**: nein
- **System**: nein
- **Optional**: ja
- **Beispielwert**: true
- **Seit**: 6.9.4.0

start.groovy.task.enabled

- **Modul**: cmas-app-admin-tool
- **Beschreibung**: Um Skripte vom Typ *Task* im Admin Tool (Navigationsgruppe *Dienste*, Navigationselement *Task-Ausführung*) ausführen zu können, ist es nötig, den Button *Start* zu aktivieren, der standardmäßig ausgeblendet ist. Setzen Sie hierfür diese System-Property auf *true*.
- **Typ**: boolean
- **Neustart erforderlich**: nein
- **System**: nein
- **Optional**: ja
- **Beispielwert**: true
- **Seit**: 6.9.4.0

task.panel.refresh.interval.seconds

- **Modul**: cmas-app-admin-tool
- **Beschreibung**: Zeit in Sekunden, nachdem die Task-Liste (im Admin Tool) des Task Execution Framework aktualisiert wird.
- **Typ**: Integer
- **Neustart erforderlich**: nein
- **System**: nein
- **Optional**: nein
- **Beispielwert**: 10
- **Seit**: 6.10.5.3 (wird beim Update von einer Version vor 6.10.5.3 nicht automatisch hinzugefügt!)

G.2.2.2 cmas-core-cache (Modul)

cache-cluster-name

- **Modul**: cmas-core-cache
- **Beschreibung**: Cache-Cluster-Name des JBoss.
- **Typ**: string
- **Neustart erforderlich**: ja
- **System**: ja
- **Optional**: nein
- **Beispielwert**: 635a6de1-629a-4129-8299-2d98633310f0
- **Seit**: 6.4.0

eviction.event.queue.size

- **Modul**: cmas-core-cache
- **Beschreibung**: Die Größe der Queue mit den Cache-Events. Der Standardwert ist 200000. Auf Systemen mit viel Traffic oder Last wird empfohlen, diesen Wert leicht zu erhöhen (bis 400000).
- **Typ**: integer
- **Neustart erforderlich**: ja
- **System**: ja
- **Optional**: nein
- **Beispielwert**: 200000
- **Seit**: 6.4.0

eviction.max.nodes

- **Modul**: cmas-core-cache
- **Beschreibung**: Setzt die maximale Größe der internen Caches. Der Standardwert ist 100000. Das Erhöhen dieses Wertes führt zu einem höheren Speicherverbrauch und wird nicht empfohlen, außer wenn ConSol explizit dazu auffordert.
- **Typ**: integer
- **Neustart erforderlich**: ja
- **System**: ja
- **Optional**: nein
- **Beispielwert**: 100000
- **Seit**: 6.4.0

eviction.wakeup.interval

- **Modul**: cmas-core-cache
- **Beschreibung**: Setzt das Intervall (in Millisekunden) zwischen zwei Verarbeitungszyklen von Cache-Queue-Events. Der Standardwert ist 3000. Auf Systemen mit viel Traffic oder Last wird empfohlen, den Wert zu verringern (Minimalwert ist 1500).
- **Typ**: integer
- **Neustart erforderlich**: ja
- **System**: ja
- **Optional**: nein
- **Beispielwert**: 3000
- **Seit**: 6.4.0

G.2.2.3 cmas-core-index-common (Modul)

big.task.minimum.size

- **Modul**: cmas-core-index-common
- **Beschreibung**: Gibt die Minimalgröße eines Index-Tasks an (in Teilen, jeder Teil hat 100 Einheiten), um diesen Task als einen großen Task zu qualifizieren. Große Tasks haben eine niedrigere Priorität als normale.
- **Typ**: integer
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: 15 (Standardwert)
- **Seit**: 6.8.3

database.notification.enabled

- **Modul**: cmas-core-index-common
- **Beschreibung**: Legt fest, ob statt JMS der Database Notification Channel der Index-Aktualisierung verwendet werden soll.
- **Typ**: boolean
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: false
- **Seit**: 6.8.4.7

database.notification.redelivery.delay.seconds

- **Modul**: cmas-core-index-common
- **Beschreibung**: Legt bei Verwendung des Database Notification Channel der Index-Aktualisierung fest, mit welcher Verzögerung die Benachrichtigung erneut gesendet wird, wenn eine Exception auftritt.
- **Typ**: integer
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: 60
- **Seit**: 6.8.4.7

database.notification.redelivery.max.attempts

- **Modul**: cmas-core-index-common
- **Beschreibung**: Legt bei Verwendung des Database Notification Channel der Index-Aktualisierung fest, wie oft maximal versucht wird, die Benachrichtigung erneut zu senden, wenn eine Exception auftritt.
- **Typ**: integer
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: 60
- **Seit**: 6.8.4.7

disable.admin.task.auto.commit

- **Modul**: cmas-core-index-common
- **Beschreibung**: Alle Tasks, die für eine Index-Aktualisierung erstellt werden, werden automatisch direkt nach ihrer Erstellung ausgeführt.
- **Typ**: boolean
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: false
- **Seit**: 6.6.1

index.attachment

- **Modul**: cmas-core-index-common
- **Beschreibung**: Beschreibt, ob der Inhalt von Attachments indiziert wird.
- **Typ**: boolean
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: true
- **Seit**: 6.4.3

index.history

- **Modul**: cmas-core-index-common
- **Beschreibung**: Legt fest, ob das Unit- und das Ticketprotokoll indiziert werden.
- **Typ**: boolean

- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: false
- **Seit**: 6.1.0
- **Entfernt in**: 6.11.0

index.status

- **Modul**: cmas-core-index-common
- **Beschreibung**: Status des Indexers, mögliche Werte sind RED, YELLOW, GREEN, werden im Admin Tool angezeigt.
- **Typ**: string
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: GREEN
- **Seit**: 6.6.1

index.task.worker.threads

- **Modul**: cmas-core-index-common
- **Beschreibung**: Legt fest, wie viele Threads benutzt werden, um Index-Aufgaben auszuführen (Synchronisierung, Administrations- und Reparatur-Aufgaben).
- **Typ**: integer
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: 1 (Standardwert) (wir empfehlen einen Wert zu verwenden, der nicht größer als 2 ist)
- **Seit**: 6.6.14, 6.7.3. Seit 6.8.0 und auschließlich in 6.6.21 sind auch normale (live) Index-Aktualisierungen von dieser Property betroffen.

index.version.current

- **Modul**: cmas-core-index-common
- **Beschreibung**: Enthält Informationen über die derzeitige (möglicherweise veraltete) Index-version.
- **Typ**: integer
- **Neustart erforderlich**: nein

- **System**: ja
- **Optional**: nein
- **Beispielwert**: 1 (Standardwert)
- **Seit**: 6.7.0

index.version.newest

- **Modul**: cmas-core-index-common
- **Beschreibung**: Enthält Informationen, welche Indexversion als die neueste betrachtet wird.
- **Typ**: integer
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: 1 (Standardwert)
- **Seit**: 6.7.0

indexed.assets.per.thread.in.memory

- **Modul**: cmas-core-index-common
- **Beschreibung**: Legt fest, wie viele Assets während des Indizierens pro Thread auf einmal in den Speicher geladen werden.
- **Typ**: integer
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: 200 (Standardwert)
- **Seit**: 6.8.0

indexed.engineers.per.thread.in.memory

- **Modul**: cmas-core-index-common
- **Beschreibung**: Legt fest, wie viele Bearbeiter während des Indizierens pro Thread auf einmal in den Speicher geladen werden.
- **Typ**: integer
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: 300 (Standardwert)
- **Seit**: 6.6.14, 6.7.3

indexed.resources.per.thread.in.memory

- **Modul**: cmas-core-index-common
- **Beschreibung**: Legt fest, wie viele Ressourcen während des Indizierens pro Thread auf einmal in den Speicher geladen werden.
- **Typ**: integer
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: 200 (Standardwert)
- **Seit**: 6.10.0.0

indexed.tickets.per.thread.in.memory

- **Modul**: cmas-core-index-common
- **Beschreibung**: Legt fest, wie viele Tickets während des Indizierens pro Thread auf einmal in den Speicher geladen werden.
- **Typ**: integer
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: 100 (Standardwert)
- **Seit**: 6.6.14, 6.7.3

indexed.units.per.thread.in.memory

- **Modul**: cmas-core-index-common
- **Beschreibung**: Legt fest, wie viele Units während des Indizierens pro Thread auf einmal in den Speicher geladen werden.
- **Typ**: integer
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: 200 (Standardwert)
- **Seit**: 6.6.14, 6.7.3

synchronize.master.address

- **Modul**: cmas-core-index-common
- **Beschreibung**: Wert der Java-System-Property *-Dcmas.http.host.port*, die angibt, unter welcher URL der Index-Master-Server erreichbar ist. Standard ist Null. Seit CM-Version 6.6.17 ist dieser Wert beim Setup konfigurierbar, um den initialen Index-Master-Server zu bestimmen. Bitte beachten Sie, dass das Verändern dieses Wertes nur erlaubt ist, wenn alle Cluster-Nodes zum Empfang von Index-Veränderungen gestoppt sind.
- **Typ**: integer
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: ja
- **Beispielwert**: 127.0.0.1:80
- **Seit**: 6.6.0

synchronize.master.security.token

- **Modul**: cmas-core-index-common
- **Beschreibung**: Das Passwort für den URL-Zugriff auf den Index-Snapshot, z. B. für die Index-Synchronisierung oder für Backups.
- **Typ**: string
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: ja
- **Beispielwert**: token
- **Seit**: 6.6.0

synchronize.master.security.user

- **Modul**: cmas-core-index-common
- **Beschreibung**: Der Benutzername für den URL-Zugriff auf den Index-Snapshot, z. B. für die Index-Synchronisierung oder für Backups.
- **Typ**: string
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: ja
- **Beispielwert**: user
- **Seit**: 6.6.0

synchronize.master.timeout.minutes

- **Modul**: cmas-core-index-common
- **Beschreibung**: Legt fest, wie oft die Index-Synchronisierung ausgehend vom aktuellen Master-Server fehlschlagen darf, bis ein neuer Master für die Index-Reparatur ausgewählt wird. Standardwert ist 5. Seit CM-Version 6.6.17 ist dieser Wert im Setup konfigurierbar, wobei 0 bedeutet, dass der Master-Server nie geändert wird (Failover-Mechanismus deaktiviert).
- **Typ**: integer
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: 5
- **Seit**: 6.6.0

synchronize.megabits.per.second

- **Modul**: cmas-core-index-common
- **Beschreibung**: Legt fest, wie viel Bandbreite der Master-Server verbrauchen darf, um Index-Änderungen an die Slave-Server zu übermitteln. Standardwert ist 85. Nutzen Sie nicht die gesamte verfügbare Bandbreite, um die Index-Änderungen zwischen den Hosts zu übermitteln, da dies dafür sorgen kann, dass die Nodes des Clusters nicht mehr synchron sind.
- **Typ**: integer
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: 85
- **Seit**: 6.6.0

synchronize.sleep.millis

- **Modul**: cmas-core-index-common
- **Beschreibung**: Legt fest, wie oft jeder Slave-Server den Master-Server auf Änderungen am Index abfragt. Standardwert ist 1000.
- **Typ**: integer
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: 1000
- **Seit**: 6.6.0

G.2.2.4 cmas-core-security (Modul)

admin.email

- **Modul**: cmas-core-security
- **Beschreibung**: Die E-Mail-Adresse des ConSol CM-Administrators. Anfänglich wird hier der Wert genommen, den Sie bei der Systemeinrichtung eingegeben haben.
- **Typ**: string
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: myuser@consol.de
- **Seit**: 6.0

admin.login

- **Modul**: cmas-core-security
- **Beschreibung**: Der Name des ConSol CM-Administrators. Anfänglich wird hier der Wert genommen, den Sie bei der Systemeinrichtung eingegeben haben.
- **Typ**: string
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: admin
- **Seit**: 6.0

authentication.method

- **Modul**: cmas-core-security
- **Description**: Methode der Bearbeiter-Authentifizierung für den Web Client (interne CM-Datenbank oder LDAP-Authentifizierung). Mögliche Werte sind LDAP oder DATABASE.
- **Typ**: string
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: DATABASE
- **Seit**: 6.0

contact.authentication.method

- **Modul**: cmas-core-security
- **Beschreibung**: Definiert die Kontakt-Authentifizierungsmethode für CM.Track, mögliche Werte sind DATABASE oder LDAP oder LDAP,DATABASE oder DATABASE,LDAP.
- **Typ**: string
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Seit**: 6.9.3.0

contact.inherit.permissions.only.to.own.customer.group

- **Modul**: cmas-core-security
- **Description**: Legt fest, ob der authentifizierte Kontakt in CM.Track alle Kundengruppen-Berechtigungen vom repräsentierenden Bearbeiter erbt (false) oder nur die Berechtigungen für die eigene Kundengruppe hat (true).
- **Typ**: boolean
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Seit**: 6.9.2.3

kerberos.v5.enabled

- **Modul**: cmas-core-security
- **Description**: Legt fest, ob SSO über Kerberos aktiviert ist.
- **Typ**: boolean
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: false (Standardwert, wenn Kerberos bei der Systemeinrichtung nicht aktiviert wurde)
- **Seit**: 6.2.0

kerberos.v5.username.regex

- **Modul**: cmas-core-security
- **Beschreibung**: Regulärer Ausdruck für die Zuordnung des Kerberos Principals zum Login des CM-Bearbeiters
- **Typ**: string

- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: (.*)@.*
- **Seit**: 6.2.0

ldap.authentication

- **Modul**: cmas-core-security
- **Description**: Verwendete Authentifizierungsmethode, wenn LDAP-Authentifizierung benutzt wird.
- **Typ**: string
- **Neustart erforderlich**: ja
- **System**: ja
- **Optional**: nein
- **Beispielwert**: simple
- **Seit**: 6.0

ldap.basedn

- **Modul**: cmas-core-security
- **Beschreibung**: Base DN für die Suche von LDAP-Benutzerkonten, wenn LDAP-Authentifizierung verwendet wird.
- **Typ**: string
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: ou=accounts,dc=consol,dc=de
- **Seit**: 6.0

ldap.contact.name.basedn

- **Modul**: cmas-core-security
- **Description**: Base DN für die Suche nach der Kontakt-DN mittels LDAP-ID (z. B. ou=accounts,dc=consol,dc=de).
- **Typ**: string
- **Neustart erforderlich**: nein
- **System**: nein
- **Optional**: ja
- **Seit**: 6.9.3.0

ldap.contact.name.password

- **Modul**: cmas-core-security
- **Beschreibung**: Passwort für die Suche nach der Kontakt-DN mittels LDAP-ID. Wenn nicht gesetzt, wird ein anonymes Konto verwendet.
- **Typ**: string
- **Neustart erforderlich**: nein
- **System**: nein
- **Optional**: ja
- **Seit**: 6.9.3.0

ldap.contact.name.providerurl

- **Modul**: cmas-core-security
- **Beschreibung**: Adresse des LDAP-Servers (ldap[s]://host:port).
- **Typ**: string
- **Neustart erforderlich**: nein
- **System**: nein
- **Optional**: ja
- **Seit**: 6.9.3.0

ldap.contact.name.searchattr

- **Modul**: cmas-core-security
- **Beschreibung**: Attribut für die Suche nach der Kontakt-DN mittels LDAP-ID (z. B. uid).
- **Typ**: string
- **Neustart erforderlich**: nein
- **System**: nein
- **Optional**: ja
- **Seit**: 6.9.3.0

ldap.contact.name.userdn

- **Modul**: cmas-core-security
- **Beschreibung**: Benutzer-DN für die Suche nach Kontakt-DN mittels LDAP-ID. Wenn nicht gesetzt, wird ein anonymes Konto verwendet.
- **Typ**: string
- **Neustart erforderlich**: nein
- **System**: nein
- **Optional**: ja
- **Seit**: 6.9.3.0

ldap.initialcontextfactory

- **Modul**: cmas-core-security
- **Beschreibung**: Name der Java-Klasse für die Initial Context Factory der LDAP-Implementierung bei der Verwendung von LDAP-Authentifizierung. Ist üblicherweise *com.-sun.jndi.ldap.LdapCtxFactory*.
- **Typ**: string
- **Neustart erforderlich**: ja
- **System**: ja
- **Optional**: nein
- **Beispielwert**: com.sun.jndi.ldap.LdapCtxFactory
- **Seit**: 6.0

ldap.password

- **Modul**: cmas-core-security
- **Beschreibung**: Passwort für die Verbindung zum LDAP, um Benutzer zu suchen, wenn LDAP-Authentifizierung verwendet wird. Wird nur benötigt, wenn die Suche nicht anonym ausgeführt werden kann.
- **Typ**: password
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: ja
- **Seit**: 6.1.2

ldap.providerurl

- **Modul**: cmas-core-security
- **Beschreibung**: LDAP-Provider, wenn LDAP-Authentifizierung verwendet wird.
- **Typ**: string
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: ldap://myserver.consol.de:389
- **Seit**: 6.0

ldap.searchattr

- **Modul**: cmas-core-security
- **Beschreibung**: Suchattribut für die Suche nach LDAP-Einträgen, die mit dem CM-Login verbunden sind.

- **Typ**: string
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: uid
- **Seit**: 6.0

ldap.userdn

- **Modul**: cmas-core-security
- **Beschreibung**: LDAP-Benutzer für die Verbindung zum LDAP, um Benutzer zu suchen, wenn LDAP-Authentifizierung verwendet wird. Wird nur benötigt, wenn die Suche nicht anonym ausgeführt werden kann.
- **Typ**: string
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: ja
- **Seit**: 6.1.2

policy.password.age

- **Modul**: cmas-core-security
- **Beschreibung**: Definiert (in Tagen), wie alt das Passwort sein darf.
- **Typ**: integer
- **Neustart erforderlich**: nein
- **System**: nein
- **Optional**: ja
- **Beispielwert**: 5500 (15 Jahre, Standardwert)
- **Seit**: 6.10.1.0

policy.password.pattern

- **Modul**: cmas-core-security
- **Beschreibung**: Definiert das Passwortmuster.
- **Typ**: string
- **Neustart erforderlich**: nein
- **System**: nein
- **Optional**: ja
- **Beispielwert**: "^.3,$" (Standardwert)
- **Seit**: 6.10.1.0

policy.rotation.ratio

- **Modul**: cmas-core-security
- **Beschreibung**: Legt fest, wie oft sich das Passwort wiederholen darf. Ist der Wert z. B. 10, darf das neue Passwort nicht unter den zehn letzten Passwörtern des Benutzers sein.
- **Typ**: integer
- **Neustart erforderlich**: nein
- **System**: nein
- **Optional**: ja
- **Beispielwert**: 1 (Standardwert)
- **Seit**: 6.10.1.0

policy.username.case.sensitive

- **Modul**: cmas-core-security
- **Beschreibung**: Legt fest, ob bei Benutzernamen die Groß- und Kleinschreibung beachtet werden muss.
- **Typ**: boolean
- **Neustart erforderlich**: nein
- **System**: nein
- **Optional**: ja
- **Beispielwert**: true (Standardwert)
- **Seit**: 6.10.1.0

resetCode.expiriationPeriod

- **Modul**: cmas-core-security
- **Beschreibung**: Definiert die Gültigkeitsdauer des Links zum Zurücksetzen des Passworts in CM.Track.
- **Typ**: Integer
- **Neustart erforderlich**: nein
- **System**: nein
- **Optional**: ja
- **Beispielwert**: 86400000 (Standardwert, 24 Stunden)
- **Seit**: 6.10.1

G.2.2.5 cmas-core-server (Modul)

attachment.allowed.types

- **Modul**: cmas-core-server
- **Beschreibung**: Durch Kommas getrennte Liste der erlaubten Dateinamenserweiterungen (wenn kein Werte definiert ist, sind alle Dateinamenserweiterungen erlaubt).
- **Typ**: string
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: ja
- **Beispielwert**: txt,zip,doc
- **Seit**: 6.5.0

attachment.max.size

- **Modul**: cmas-core-server
- **Beschreibung**: Maximale Größe von Attachments in MB. Dies ist eine Validierungs-Property der CM-API. Sie steuert die Größe der Attachments in Tickets, Units und Ressourcen. Außerdem steuert sie die Größe der eingehenden (nicht ausgehenden!) E-Mail-Attachments im NIMH- sowie im Mule/ESB-Modus.
- **Typ**: integer
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: 100
- **Seit**: 6.4.0

config.data.version

- **Modul**: cmas-core-server
- **Beschreibung**: Die interne Versionsnummer der aktuellen Systemkonfiguration. Diese Property wird intern gepflegt. Ändern Sie sie nur, wenn Sie von ConSol dazu aufgefordert werden.
- **Typ**: integer
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: 11
- **Seit**: 6.0

defaultCommentClassName

- **Modul**: cmas-core-server
- **Beschreibung**: Standard-Textklasse für Kommentare.
- **Typ**: string
- **Neustart erforderlich**: nein
- **System**: nein
- **Optional**: ja
- **Beispielwert**:
- **Seit**: 6.3.0

defaultIncommingMailClassName

- **Modul**: cmas-core-server
- **Beschreibung**: Standard-Textklasse für eingehende E-Mails.
- **Typ**: string
- **Neustart erforderlich**: nein
- **System**: nein
- **Optional**: ja
- **Seit**: 6.3.0

defaultOutgoingMailClassName

- **Modul**: cmas-core-server
- **Beschreibung**: Standard-Textklasse für ausgehende E-Mails.
- **Typ**: string
- **Neustart erforderlich**: nein
- **System**: nein
- **Optional**: ja
- **Beispielwert**:
- **Seit**: 6.3.0

fetchSize.strategy

- **Modul**: cmas-core-server
- **Beschreibung**: Auswahl der Strategie für die Abholgröße von JDBC-Ergebnissatzes.
- **Typ**: string
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: ja

- **Beispielwert**: FetchSizePageBasedStrategy, FetchSizeThresholdStrategy, FetchSizeFixedStrategy
- **Seit**: 6.8.4.1

fetchSize.strategy.FetchSizeFixedStrategy.value

- **Modul**: cmas-core-server
- **Beschreibung**: Legt den Wert für Abholgrößen fest, wenn die ausgewählte Strategie für die Abholgröße *FetchSizeFixedStrategy* ist.
- **Typ**: integer
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: ja
- **Beispielwert**: 150
- **Seit**: 6.8.4.1

fetchSize.strategy.FetchSizePageBasedStrategy.limit

- **Modul**: cmas-core-server
- **Beschreibung**: Legt den Wert für Abholgrößen fest, wenn die ausgewählte Strategie für die Abholgröße *FetchSizePageBasedStrategy* ist.
- **Typ**: integer
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: ja
- **Beispielwert**: 10000
- **Seit**: 6.8.4.1

fetchSize.strategy.FetchSizeThresholdStrategy.value

- **Modul**: cmas-core-server
- **Beschreibung**: Gibt Grenzwerte für Abholgrößen an, wenn die ausgewählte Strategie für die Abholgröße *FetchSizeThresholdStrategy* ist.
- **Typ**: integer
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: ja
- **Beispielwert**: 150,300,600,1000
- **Seit**: 6.8.4.1

last.config.change

- **Modul**: cmas-core-server
- **Beschreibung**: Zufällige UUID, die während der letzten Konfigurationsänderung generiert wurde.
- **Typ**: string
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: 2573c7b7-2bf5-47ff-b5a2-bad31951a266
- **Seit**: 6.1.0, 6.2.1

ldap.certificate.basedn

- **Modul**: cmas-core-server
- **Beschreibung**: Base DN für den Speicherort der Zertifikate im LDAP-Verzeichnisbaum. Wenn nicht angegeben, wird *cmas-core-security, ldap.basedn* verwendet.
- **Typ**: string
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: ja
- **Beispielwert**: ou=accounts,dc=consol,dc=de
- **Seit**: 6.8.4

ldap.certificate.content.attribute

- **Modul**: cmas-core-server
- **Description**: Name des LDAP-Attributs, das angibt, wo Zertifikatsdaten im LDAP-Verzeichnisbaum gespeichert werden. Standardwert: usercertificate
- **Typ**: string
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: ja
- **Beispielwert**: usercertificate
- **Seit**: 6.8.4

ldap.certificate.password

- **Modul**: cmas-core-server
- **Beschreibung**: Passwort des LDAP-Zertifikatmanagers. Wenn nicht gesetzt, wird *cmas-core-security, ldap.password* verwendet.

- **Typ**: string
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: ja
- **Seit**: 6.8.4

ldap.certificate.providerurl

- **Modul**: cmas-core-server
- **Beschreibung**: URL des LDAP-Zertifikat-Providers. Wenn nicht gesetzt, wird *cmas-core-security, ldap.providerurl* verwendet.
- **Typ**: string
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: ja
- **Beispielwert**: ldap://ldap.consol.de:389
- **Seit**: 6.8.4

ldap.certificate.searchattr

- **Modul**: cmas-core-server
- **Description**: Name des LDAP-Attributs, mit dem Zertifikate im LDAP-Verzeichnisbaum gesucht werden. Standardwert: mail
- **Typ**: string
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: ja
- **Beispielwert**: mail
- **Seit**: 6.8.4

ldap.certificate.userdn

- **Modul**: cmas-core-server
- **Beschreibung**: DN des LDAP-Zertifikatmanagers. Wenn nicht gesetzt, wird *cmas-core-security, ldap.userdn* verwendet.
- **Typ**: string
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: ja
- **Seit**: 6.8.4

mail.encryption

- **Modul**: cmas-core-server
- **Beschreibung**: Wenn dieser Wert auf *true* gesetzt ist, ist im Ticket-E-Mail-Editor die Checkbox zur Verschlüsselung der E-Mail standardmäßig aktiviert.
- **Typ**: boolean
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: true (Standardwert = false)
- **Seit**: 6.8.4.0

mail.notification.engineerChange

- **Modul**: cmas-core-server
- **Description**: Legt fest, ob eine Benachrichtigungs-E-Mail verschickt wird, wenn der Bearbeiter eines Tickets wechselt.
- **Typ**: boolean
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: true
- **Seit**: 6.1.0

mail.notification.sender

- **Modul**: cmas-core-server
- **Beschreibung**: From-Adresse der Benachrichtigungs-E-Mails, die verschickt werden, wenn der Bearbeiter eines Tickets wechselt. Wenn nicht gesetzt, wird stattdessen *cmas-core-security, admin.email* verwendet.
- **Typ**: string
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: ja
- **Beispielwert**: cm6notification@cm6installation
- **Seit**: 6.6.3

mail.smtp.email

- **Modul**: cmas-core-server
- **Beschreibung**: SMTP-E-Mail-URL für ausgehende E-Mails.

- **Typ**: string
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: smtp://mail.mydomain.com:25
- **Seit**: 6.0

mail.smtp.envelopesender

- **Modul**: cmas-core-server
- **Beschreibung**: E-Mail-Adresse, die als Absender im SMTP-Envelope benutzt wird. Wenn nicht gesetzt, wird die From-Adresse der E-Mail benutzt.
- **Typ**: string
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: mysender@mydomain.com
- **Seit**: 6.5.7

max.licences.perUser

- **Modul**: cmas-core-server
- **Beschreibung**: Setzt die maximale Anzahl an Lizenzen, die ein einzelner Benutzer benutzen kann (z. B. durch Anmelden in einem anderen Browser). Standardmäßig ist dieser Wert nicht beschränkt.
- **Typ**: integer
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: ja
- **Beispielwert**: 10
- **Seit**: 6.8.4.5

monitoring.engineer.login

- **Modul**: cmas-core-server
- **Beschreibung**: Login des Monitoring-Bearbeiters.
- **Typ**: string
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: ja

- **Beispielwert**: nagios
- **Seit**: 6.9.3.0

monitoring.unit.login

- **Modul**: cmas-core-server
- **Beschreibung**: Login der Monitoring-Unit.
- **Typ**: string
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: ja
- **Beispielwert**: nagios
- **Seit**: 6.9.3.0

nimh.enabled

- **Modul**: cmas-core-server
- **Beschreibung**: Aktiviert den NIMH-Dienst. Im Cluster muss die Node-ID angehängt werden, z. B. *nimh.enabled.NODEID = true*.
- **Typ**: boolean
- **Neustart erforderlich**: nein
- **System**: nein
- **Optional**: ja
- **Beispielwert**: false
- **Seit**: 6.9.4.0

resource.replace.batchSize

- **Modul**: cmas-core-server
- **Beschreibung**: Legt die Anzahl der Objekte fest, die in einer Ressourcen-Ersetzen-Aktion verarbeitet werden.
- **Typ**: integer
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: 5
- **Seit**: 6.10.0.0

resource.replace.timeout

- **Modul**: cmas-core-server
- **Beschreibung**: Transaktions-Timeout (in Sekunden) für einen Schritt einer Ressourcen-Ersetzen-Aktion.
- **Typ**: integer
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: 120
- **Seit**: 6.10.0.0

script.logging.threshold.seconds

- **Modul**: cmas-core-server
- **Beschreibung**: Legt die Zeit (in Sekunden) fest, nach deren Ablauf während einer Skript-Ausführung eine Warnung in der Log-Datei generiert wird.
- **Typ**: integer
- **Neustart erforderlich**: nein
- **System**: nein
- **Optional**: ja
- **Beispielwert**: 10 (Standardwert)
- **Seit**: 6.10.1.0

server.session.archive.reaper.interval

- **Modul**: cmas-core-server
- **Beschreibung**: Reaper-Intervall (in Sekunden) von archivierten Server-Sessions.
- **Typ**: integer
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: ja
- **Beispielwert**: 60
- **Seit**: 6.7.1

server.session.archive.timeout

- **Modul**: cmas-core-server
- **Beschreibung**: Timeout der Gültigkeit der Server-Session-Archive (in Tagen). Nach diesem Zeitraum werden die Informationen zur Session aus der Datenbank entfernt.
- **Typ**: integer

- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: 31
- **Seit**: 6.7.1

server.session.reaper.interval

- **Modul**: cmas-core-server
- **Beschreibung**: Interval (in Sekunden), in dem der sog. Reaper inaktive (= beendete) Server-Sessions löscht.
- **Typ**: integer
- **Neustart erforderlich**: Nur Session Service
- **System**: ja
- **Optional**: nein
- **Beispielwert**: 60
- **Seit**: 6.6.1, 6.7.1

server.session.timeout

- **Modul**: cmas-core-server
- **Beschreibung**: Server-Session-Timeout (in Sekunden) für verbundene Clients. Jeder Client kann dieses Timeout mit benutzerdefinierten Werten mittels seiner ID (ADMIN_TOOL, WEB_CLIENT, WORKFLOW_EDITOR, TRACK (vor 6.8 bitte PORTER verwenden), ETL, REST) überschreiben. Diese wird an den Namen der System-Property angehängt, z. B. *server.session.timeout.ADMIN_TOOL*.
  Siehe auch die Attribute der Seitenanpassung *updateTimeServerSessionActivityEnabled* und *updateTimeServerSessionActivity*, beide vom Typ *cmApplicationCustomization*.
- **Typ**: integer
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: 1800
- **Seit**: 6.6.1, 6.7.1

Detaillierte Erklärung für das Admin Tool:

- server.session.timeout.ADMIN_TOOL
  Definiert den Zeitraum, den der Server eine Session im Admin Tool als gültig betrachtet, wenn keine Aktivität im Admin Tool der Session erfolgt. Das Admin Tool kennt diesen Wert nicht, es bemerkt nur eine ungültige Session, wenn es länger keine Aktivität gegeben hat.

- admin.tool.session.check.interval
  Definiert den Zeitraum zwischen zwei Überprüfungen durch das Admin Tool, ob der Server die Session noch als gültig betrachtet.

Wenn zum Beispiel admin.tool.session.check.interval = 60, fragt das Admin Tool den Server einmal pro Minute, ob die Session noch aktiv/gültig ist. Wenn server.session.timeout.ADMIN_TOOL = 600 erhält das Admin Tool die Antwort, dass die Session ungültig ist, nach zehn Minuten der Inaktivität.

skip.wfl.transfer.cleanup

- **Modul**: cmas-core-server
- **Beschreibung**: Wenn diese System-Property auf *true* gesetzt wird, wird das Workflow-Cleanup nach der Übertragung übersprungen.
- **Typ**: boolean
- **Neustart erforderlich**: nein
- **System**: nein
- **Optional**: ja
- **Beispielwert**: false (Standardwert)
- **Seit**: 6.9.4.1

tickets.delete.size

- **Modul**: cmas-core-server
- **Beschreibung**: Definiert die Anzahl der Tickets, die pro Transaktion gelöscht werden. Standardmäßig ist dieser Wert 10.
- **Typ**: integer
- **Neustart erforderlich**: Nur Session Service
- **System**: ja
- **Optional**: nein
- **Beispielwert**: 10
- **Seit**: 6.8.1

ticket.delete.timeout

- **Modul**: cmas-core-server
- **Beschreibung**: Transaktions-Timeout (in Sekunden) beim Löschen von Tickets.
- **Typ**: integer
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: 60
- **Seit**: 6.1.3

transaction.timeout.minutes

- **Modul**: cmas-core-server
- **Beschreibung**: Setzt den Timeout für den Task Execution Service des TEF, d. h. ein Durchlauf eines Tasks muss vor dem Ablauf dieser Zeitspanne abgeschlossen sein. Die Änderungen sind nur für neue Aufgaben sichtbar, deren Ausführung nach der Konfigurationsänderung beginnt.
- **Typ**: integer
- **Neustart erforderlich**: nein
- **System**: nein
- **Optional**: ja
- **Beispielwert**: 10*3600 (10 Stunden - Standardwert)
- **Seit**: 6.10

unit.replace.batchSize

- **Modul**: cmas-core-server
- **Beschreibung**: Legt die Anzahl der Objekte fest, die in einer Unit-Ersetzen-Aktion verarbeitet werden.
- **Typ**: integer
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: 5
- **Seit**: 6.8.2

unit.replace.timeout

- **Modul**: cmas-core-server
- **Beschreibung**: Transaktions-Timeout (in Sekunden) für einen Schritt einer Unit-Ersetzen-Aktion.
- **Typ**: integer
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: 120
- **Seit**: 6.8.2

unused.content.remover.cluster.node.id

- **Modul**: cmas-core-server
- **Beschreibung**: Wert eines *cmas.clusternode.id*, der angibt, welcher Node nicht verwendete Ticket-Attachments und Unit-Inhaltseinträge entfernt.

- **Typ**: string
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: ja
- **Beispielwert**: 1 (in der Annahme, dass der Cluster-Node mit dem Parameter *-Dcmas.clusternode.id=1* gestartet ist)
- **Seit**: 6.9.0.0

unused.content.remover.enabled

- **Modul**: cmas-core-server
- **Beschreibung**: Legt fest, ob das Entfernen ungenutzter Ticket-Attachments und Unit-Inhaltseinträge durchgeführt werden soll.
- **Typ**: boolean
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: true
- **Seit**: 6.9.0.0

unused.content.remover.polling.minutes

- **Modul**: cmas-core-server
- **Beschreibung**: Legt fest, wie oft überprüft werden soll, ob ungenutzte Ticket-Attachments und Unit-Inhaltseinträge zum Entfernen vorhanden sind.
- **Typ**: integer
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: 15
- **Seit**: 6.9.0.0

unused.content.remover.ttl.minutes

- **Modul**: cmas-core-server
- **Beschreibung**: Mindestzeitraum, nach dem ungenutzte Ticket-Attachments und Unit-Inhaltseinträge entfernt werden können.
- **Typ**: integer
- **Neustart erforderlich**: nein
- **System**: ja

- **Optional**: nein
- **Beispielwert**: 1440
- **Seit**: 6.9.0.0

warmup.executor.enabled

- **Modul**: cmas-core-server
- **Beschreibung**: Legt fest, ob der Server beim Start asynchron anlaufen soll und nebenher andere Aufgaben erledigt (z. B. interne Caches füllen).
- **Typ**: boolean
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: true
- **Seit**: 6.9.4.2

G.2.2.6 cmas-core-shared (Modul)

cluster.mode

- **Modul**: cmas-core-shared
- **Beschreibung**: Legt fest, ob CMAS in einem Cluster läuft.
- **Typ**: boolean
- **Neustart erforderlich**: ja
- **System**: ja
- **Optional**: nein
- **Beispielwert**: false
- **Seit**: 6.1.0

data.directory

- **Modul**: cmas-core-shared
- **Beschreibung**: Verzeichnis für die CMAS-Daten (z. B. Index)
- **Typ**: string
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: C:\Users\user\cmas
- **Seit**: 6.0

G.2.2.7 cmas-dwh-server (Modul)

autocommit.cf.changes

- **Modul**: cmas-dwh-server
- **Beschreibung**: Definiert, ob DWH-Aufgaben, die aufgrund von Konfigurationsänderungen an Benutzerdefinierten Feldern anfallen, automatisch ohne manuelle Interaktion im Admin Tool ausgeführt werden. Diese Property kann auch im Admin Tool im Navigationselement *DWH* gesetzt werden. Der Standardwert und empfohlene Wert ist *false*.
- **Typ**: boolean
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: false
- **Seit**: 6.7.0

batch-commit-interval

- **Modul**: cmas-dwh-server
- **Beschreibung**: Anzahl der Objekte in einer JMS-Nachricht. Höhere Werte bedeuten eine bessere Übertragungsperformance und größeren Speicherverbrauch.
- **Typ**: integer
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: ja
- **Beispielwert**: 100
- **Seit**: 6.0.0

communication.channel

- **Modul**: cmas-dwh-server
- **Beschreibung**: Kommunikationskanal, mögliche Werte sind DIRECT (Datenbank-Kommunikationskanal, Standardwert seit 6.9.4.1) oder JMS (Standardwert vor 6.9.4.1). Diese System-Property muss vor 6.9.4.1 extra hinzugefügt werden.
- **Typ**: string
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: DIRECT
- **Seit**: 6.8.5.0
- **Entfernt in**: 6.11.0.0

dwh.mode

- **Modul**: cmas-dwh-server
- **Beschreibung**: Aktueller Modus der DWH-Datenübermittlung. Mögliche Werte sind OFF, ADMIN, LIVE.
- **Typ**: string
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: OFF
- **Seit**: 6.0.1

ignore-queues

- **Modul**: cmas-dwh-server
- **Beschreibung**: Durch eine durch Kommas getrennte Liste von Queue-Namen wird hier festgelegt, dass Tickets dieser Queues nicht ins DWH übermittelt werden.
- **Typ**: string
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: ja
- **Beispielwert**: QueueName1,QueueName2,QueueName3
- **Seit**: 6.6.19
- **Entfernt in**: 6.8.1

is.cmrf.alive

- **Modul**: cmas-dwh-server
- **Beschreibung**: Als Startpunkt sollte die Zeit genommen werden, bei der zuletzt eine Nachricht an das CMRF gesendet wurde. Wenn nach diesem Wert (in Sekunden) keine Antwort vom CMRF empfangen wird, wird ein DWH-Betriebsstatus mit der Fehlermeldung, dass das CMRF nicht erreichbar ist, erstellt.
- **Typ**: integer
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: 1200
- **Seit**: 6.7.0

java.naming.factory.initial

- **Modul**: cmas-dwh-server
- **Beschreibung**: Factory-Java-Klasse für DWH context factory.
- **Typ**: string
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: org.jnp.interfaces.NamingContextFactory
- **Seit**: 6.0.1
- **Entfernt in**: 6.11.0.0

java.naming.factory.url.pkgs

- **Modul**: cmas-dwh-server
- **Beschreibung**:
- **Typ**: string
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: org.jboss.naming:org.jnp.interfaces
- **Seit**: 6.0.1
- **Entfernt in**: 6.11.0.0

java.naming.provider.url

- **Modul**: cmas-dwh-server
- **Beschreibung**: URL des Naming Provider.
- **Typ**: string
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: localhost
- **Seit**: 6.0.1
- **Entfernt in**: 6.11.0.0

notification.error.description

- **Modul**: cmas-dwh-server
- **Beschreibung**: Text für E-Mails mit Fehlermeldungen des DWH.

- **Typ**: string
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: Fehler aufgetreten
- **Seit**: 6.0.1

notification.error.from

- **Modul**: cmas-dwh-server
- **Beschreibung**: From-Adresse für E-Mails mit Fehlermeldungen des DWH.
- **Typ**: string
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: ja
- **Seit**: 6.0.1

notification.error.subject

- **Modul**: cmas-dwh-server
- **Beschreibung**: Betreff für E-Mails mit Fehlermeldungen des DWH.
- **Typ**: string
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: Fehler aufgetreten
- **Seit**: 6.0.1

notification.error.to

- **Modul**: cmas-dwh-server
- **Beschreibung**: To-Adresse für E-Mails mit Fehlermeldungen des DWH.
- **Typ**: string
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: myuser@consol.de
- **Seit**: 6.0.1

notification.finished_successfully.description

- **Modul**: cmas-dwh-server
- **Beschreibung**: Text für E-Mails des DWHs, wenn eine Übertragung erfolgreich beendet wurde.
- **Typ**: string
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: Übertragung erfolgreich beendet
- **Seit**: 6.0.1

notification.finished_successfully.from

- **Modul**: cmas-dwh-server
- **Beschreibung**: From-Adresse für E-Mails des DWHs, wenn eine Übertragung erfolgreich beendet wurde.
- **Typ**: string
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: ja
- **Seit**: 6.0.1

notification.finished_successfully.subject

- **Modul**: cmas-dwh-server
- **Beschreibung**: Betreff für E-Mails des DWHs, wenn eine Übertragung erfolgreich beendet wurde.
- **Typ**: string
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: Übertragung erfolgreich beendet
- **Seit**: 6.0.1

notification.finished_successfully.to

- **Modul**: cmas-dwh-server
- **Beschreibung**: To-Adresse für E-Mails des DWHs, wenn eine Übertragung erfolgreich beendet wurde.
- **Typ**: string
- **Neustart erforderlich**: ja

- **System**: ja
- **Optional**: nein
- **Beispielwert**: myuser@consol.de
- **Seit**: 6.0.1

notification.finished_unsuccessfully.description

- **Modul**: cmas-dwh-server
- **Beschreibung**: Text für E-Mails des DWHs, wenn eine Übertragung nicht erfolgreich beendet wurde.
- **Typ**: string
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: Übertragung nicht erfolgreich beendet
- **Seit**: 6.0.1

notification.finished_unsuccessfully.from

- **Modul**: cmas-dwh-server
- **Beschreibung**: From-Adresse für E-Mails des DWHs, wenn eine Übertragung nicht erfolgreich beendet wurde.
- **Typ**: string
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: ja
- **Seit**: 6.0.1

notification.finished_unsuccessfully.subject

- **Modul**: cmas-dwh-server
- **Beschreibung**: Betreff für E-Mails des DWHs, wenn eine Übertragung nicht erfolgreich beendet wurde.
- **Typ**: string
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: Übertragung nicht erfolgreich beendet
- **Seit**: 6.0.1

notification.finished_unsuccessfully.to

- **Modul**: cmas-dwh-server
- **Beschreibung**: To-Adresse für E-Mails des DWHs, wenn eine Übertragung nicht erfolgreich beendet wurde.
- **Typ**: string
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: myuser@consol.de
- **Seit**: 6.0.1

notification.host

- **Modul**: cmas-dwh-server
- **Description**: Hostname des E-Mail-Servers (SMTP) für das Senden von DWH-E-Mails.
- **Typ**: string
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: ja
- **Beispielwert**: myserver.consol.de
- **Seit**: 6.0.1

notification.password

- **Modul**: cmas-dwh-server
- **Beschreibung**: Passwort für das Senden von DWH-E-Mails. (optional)
- **Typ**: string
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: ja
- **Seit**: 6.0.1

notification.port

- **Modul**: cmas-dwh-server
- **Beschreibung**: SMTP-Port für das Senden von DWH-E-Mails.
- **Typ**: string
- **Neustart erforderlich**: nein
- **System**: ja

- **Optional**: ja
- **Beispielwert**: 25
- **Seit**: 6.0.1

notification.protocol

- **Modul**: cmas-dwh-server
- **Beschreibung**: Das Protokoll, das für das Senden von E-Mails aus dem DWH verwendet wird.
- **Typ**: string
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: ja
- **Beispielwert**: pop3\

notification.username

- **Modul**: cmas-dwh-server
- **Beschreibung**: Benutzername (SMTP) für das Senden von DWH-E-Mails.
- **Typ**: string
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: ja
- **Beispielwert**: myuser
- **Seit**: 6.0.1

skip-ticket

- **Modul**: cmas-dwh-server
- **Beschreibung**: Tickets werden bei der Übertragung/Aktualisierung nicht übermittelt.
- **Typ**: boolean
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: false
- **Seit**: 6.6.19
- **Entfernt in**: 6.8.1

skip-ticket-history

- **Modul**: cmas-dwh-server
- **Beschreibung**: Ticketprotokoll wird bei der Übertragung/Aktualisierung nicht übermittelt.

- **Typ**: boolean
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: false
- **Seit**: 6.6.19
- **Entfernt in**: 6.8.1

skip-unit

- **Modul**: cmas-dwh-server
- **Beschreibung**: Units werden bei der Übertragung/Aktualisierung nicht übermittelt.
- **Typ**: boolean
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: false
- **Seit**: 6.6.19
- **Entfernt in**: 6.8.1

skip-unit-history

- **Modul**: cmas-dwh-server
- **Beschreibung**: Unit-Protokoll wird bei der Übertragung/Aktualisierung nicht übermittelt.
- **Typ**: boolean
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: false
- **Seit**: 6.6.19
- **Entfernt in**: 6.8.1

split.history

- **Modul**: cmas-dwh-server
- **Beschreibung**: Ändert das SQL-Statement dahingehend, dass Ticketprotokolle während der Übertragung an das DWH nicht für alle Tickets auf einmal abgeholt werden, sondern für ein Ticket pro SQL-Statement.
- **Typ**: boolean
- **Neustart erforderlich**: nein

- **System**: ja
- **Optional**: ja
- **Beispielwert**: false
- **Seit**: 6.8.0

unit.transfer.order

- **Modul**: cmas-dwh-server
- **Beschreibung**: Legt fest, in welcher Reihenfolge Datenobjektgruppen an das DWH übertragen werden.
- **Typ**: string
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: ja
- **Beispielwert**: company;customer
- **Seit**: 6.6.19
- **Entfernt in**: 6.8.1

G.2.2.8 cmas-esb-core (Modul)

esb.directory

- **Modul**: cmas-esb-core
- **Beschreibung**: Von Mule/ESB verwendetes Verzeichnis.
- **Typ**: string
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: C:\Users\user\cmas\mule
- **Seit**: 6.0
- **Entfernt in**: 6.11.0

G.2.2.9 cmas-esb-mail (Modul)

mail.attachments.validation.info.sender

- **Modul**: cmas-esb-mail
- **Beschreibung**: Setzt den From-Header bei Attachments vom Typ *error notification mail*. Standardmäßig wird die E-Mail-Adresse des Administrators verwendet, die bei der Systeminstallation angegeben wurde.
- **Typ**: string

- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: admin@consolcm.com
- **Seit**: 6.7.5
- **Entfernt in**: 6.11.0

mail.attachments.validation.info.subject

- **Modul**: cmas-esb-mail
- **Beschreibung**: Setzt den Betreff bei Attachments vom Typ *error notification mail*.
- **Typ**: string
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: E-Mail konnte nicht verarbeitet werden, weil ihre Attachments zurückgewiesen wurden!
- **Seit**: 6.7.5
- **Entfernt in**: 6.11.0

mail.callname.pattern

- **Modul**: cmas-esb-mail
- **Beschreibung**: Regulärer Ausdruck für den Betreff von eingehenden E-Mails. Verfügbar als TICKET_NAME_PATTERN_FORMAT in Skripten für eingehende E-Mails.
- **Typ**: string
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: .*?Ticket\s+\((\S+)\).*
- **Seit**: 6.0
- **Entfernt in**: 6.11.0

mail.cluster.node.id

- **Modul**: cmas-esb-mail
- **Beschreibung**: Nur der Node, dessen *mail.cluster.node.id* gleich *cmas.clusternode.id* ist, startet den Mule/ESB E-Mail-Service.
- **Typ**: string
- **Neustart erforderlich**: nein

- **System**: ja
- **Optional**: nein
- **Beispielwert**: unspecified
- **Seit**: 6.6.5
- **Entfernt in**: 6.11.0

mail.db.archive

- **Modul**: cmas-esb-mail
- **Beschreibung**: Wenn dieser Wert auf *true* gesetzt ist, werden eingehende E-Mails in der Datenbank archiviert.
- **Typ**: boolean
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: ja
- **Beispielwert**: false (Standardwert)
- **Seit**: 6.8.5.5
- **Entfernt in**: 6.11.0

Obsolet! Im Mule/ESB-Modus werden keine E-Mails in der Datenbank gespeichert. E-Mails, die nicht verarbeitet werden können, werden im Dateisystem gespeichert, siehe Abschnitt *E-Mail-Backups* im *ConSol CM Administratorhandbuch*.

mail.delete.read

- **Modul**: cmas-esb-mail
- **Beschreibung**: Legt fest, ob CM die per IMAP(S) abgeholten E-Mails löscht. Wenn der Wert auf *true* gesetzt wird, werden die E-Mails nach der Abholung gelöscht. Standardmäßig werden die per IMAP(S) abgeholten E-Mails nicht gelöscht. Hinweis: E-Mails, die per POP3(S) abgeholt werden, werden immer gelöscht.
- **Typ**: boolean
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: true
- **Seit**: 6.7.3
- **Entfernt in**: 6.11.0

mail.incoming.uri

- **Modul**: cmas-esb-mail
- **Beschreibung**: URL für eingehende E-Mails.

- **Typ**: string
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: pop3://cm-incoming-user:password@localhost:10110
- **Seit**: 6.0
- **Entfernt in**: 6.11.0

Dieser Wert sollte nicht innerhalb des System-Property-Fensters verändert werden. Die Postfächer sollten stattdessen im Admin Tool im Navigationselement *E-Mail* konfiguriert werden. Wenn Sie dieses Element für die Konfiguration benutzen, können Sie alle Einträge kontrolliert konfigurieren, d. h. für jedes Postfach, das hinzugefügt wird, baut CM während der Einrichtung eine Testverbindung auf. Auf diese Weise ist es nicht möglich, falsche Werte einzugeben.

mail.max.restarts

- **Modul**: cmas-esb-mail
- **Beschreibung**: Maximale Anzahl der Neustarts des E-Mail-Services, bevor aufgegeben wird.
- **Typ**: integer
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: 3
- **Seit**: 6.0
- **Entfernt in**: 6.11.0

mail.mime.strict

- **Modul**: cmas-esb-mail
- **Beschreibung**: Wenn dieser Wert auf *false* gesetzt ist, werden E-Mail-Adressen nicht auf strikte MIME-Übereinstimmung geparst. Standard ist *true*, was bedeutet, dass auf strikte MIME-Übereinstimmung geprüft wird.
- **Typ**: boolean
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: false

- **Seit**: 6.6.17, 6.7.3
- **Entfernt in**: 6.11.0

mail.mule.service

- **Modul**: cmas-esb-mail
- **Beschreibung**: From-Adresse für E-Mails, die vom Mule-Service aus gesendet werden.
- **Typ**: email
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: myuser@consol.de
- **Seit**: 6.0
- **Entfernt in**: 6.11.0

mail.polling.interval

- **Modul**: cmas-esb-mail
- **Beschreibung**: Abrufintervall für E-Mails in Millisekunden.
- **Typ**: integer
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: 60000
- **Seit**: 6.0
- **Entfernt in**: 6.11.0

mail.process.error

- **Modul**: cmas-esb-mail
- **Beschreibung**: To-Adresse für E-Mails mit Fehlermeldungen des Mule. Standardmäßig wird die E-Mail-Adresse des Administrators verwendet, die bei der Systeminstallation angegeben wurde.
- **Typ**: email
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: myuser@consol.de
- **Seit**: 6.0
- **Entfernt in**: 6.11.0

mail.process.retry.attempts

- **Modul**: cmas-esb-mail
- **Beschreibung**: Anzahl der erneuten Versuche, wenn E-Mails verarbeitet werden.
- **Typ**: integer
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: 3
- **Seit**: 6.0.2
- **Entfernt in**: 6.11.0

mail.process.timeout

- **Modul**: cmas-esb-mail
- **Beschreibung**: Timeout für die E-Mail-Verarbeitung in Sekunden.
- **Typ**: integer
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: 60
- **Seit**: 6.1.3
- **Entfernt in**: 6.11.0

mail.redelivery.retry.count

- **Modul**: cmas-esb-mail
- **Beschreibung**: Gibt die Anzahl der Versuche an, eine E-Mail aus dem CM-System erneut zuzustellen.
- **Typ**: integer
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: 3
- **Seit**: 6.1.0
- **Entfernt in**: 6.11.0

G.2.2.10 cmas-nimh (Modul)

filesystem.polling.threads.number

- **Modul**: cmas-nimh
- **Beschreibung**: Anzahl der Threads, die für die Abfrage der Datenbank-E-Mail-Warteschlange gestartet werden. Standardwert: 1.
- **Typ**: integer
- **Neustart erforderlich**: nein
- **System**: nein
- **Optional**: ja
- **Beispielwert**: 10
- **Seit**: 6.4.0

filesystem.polling.threads.shutdown.timeout.seconds

- **Modul**: cmas-nimh
- **Beschreibung**: Wartezeit nach dem Shutdown-Signal. Wenn der Timeout erreicht ist, wird der Thread beendet. Standardwert: 60.
- **Typ**: integer
- **Neustart erforderlich**: nein
- **System**: nein
- **Optional**: ja
- **Beispielwert**: 60
- **Seit**: 6.4.0

filesystem.polling.threads.watchdog.interval.seconds

- **Modul**: cmas-nimh
- **Beschreibung**: Intervall des Watchdog Threads. Standardwert: 30.
- **Typ**: integer
- **Neustart erforderlich**: nein
- **System**: nein
- **Optional**: ja
- **Beispielwert**: 60
- **Seit**: 6.4.0

filesystem.task.enabled

- **Modul**: cmas-nimh
- **Beschreibung**: Mit dieser System-Property kann der Service Thread eines bestimmten Pollers deaktiviert werden. Standardwert: true.
- **Typ**: boolean
- **Neustart erforderlich**: nein
- **System**: nein
- **Optional**: ja
- **Beispielwert**: true
- **Seit**: 6.4.0

filesystem.task.interval.seconds

- **Modul**: cmas-nimh
- **Beschreibung**: Standard-Intervall für den Abruf von Postfächern. Standardwert: 60 Sekunden
- **Typ**: integer
- **Neustart erforderlich**: nein
- **System**: nein
- **Optional**: ja
- **Beispielwert**: 60
- **Seit**: 6.4.0

filesystem.task.polling.folder

- **Modul**: cmas-nimh
- **Beschreibung**: Speicherort des Ordners, der überwacht und nach E-Mails im Format von eml-Dateien durchsucht wird. Standard: Unterverzeichnis "mail" des cmas-Datenverzeichnisses
- **Typ**: string
- **Neustart erforderlich**: nein
- **System**: nein
- **Optional**: ja
- **Beispielwert**: c://cmas//mail
- **Seit**: 6.4.0

filesystem.task.timeout.seconds

- **Modul**: cmas-nimh
- **Beschreibung**: Nach dieser Zeit (der Inaktivität) wird der Service Thread als beschädigt betrachtet und automatisch neu gestartet. Standardwert: 120 Sekunden
- **Typ**: integer

- **Neustart erforderlich**: nein
- **System**: nein
- **Optional**: ja
- **Beispielwert**: 60
- **Seit**: 6.4.0

filesystem.task.transaction.timeout.seconds

- **Modul**: cmas-nimh
- **Beschreibung**: Standard-Transaktions-Timeout für Transaktionen, die E-Mails abrufen. Sollte mit der Anzahl der Nachrichten, die gleichzeitig abgeholt werden, korrelieren. Standardwert: 60 Sekunden
- **Typ**: integer
- **Neustart erforderlich**: nein
- **System**: nein
- **Optional**: ja
- **Beispielwert**: 60
- **Seit**: 6.4.0

mailbox.1.connection.host

- **Modul**: cmas-nimh
- **Beschreibung**: Host (Server) für das erste konfigurierte Postfach. Überschreibt den Standardparameter *mailbox.default.connection.host*.

mailbox.1.connection.password

- **Modul**: cmas-nimh
- **Beschreibung**: Passwort für das erste konfigurierte Postfach. Überschreibt den Standardparameter *mailbox.default.connection.password*.

mailbox.1.connection.port

- **Modul**: cmas-nimh
- **Beschreibung**: Port für das erste konfigurierte Postfach. Überschreibt den Standardparameter *mailbox.default.connection.port*.

mailbox.1.connection.protocol

- **Modul**: cmas-nimh
- **Beschreibung**: Protokoll (z. B. IMAP oder POP3) für das erste konfigurierte Postfach. Überschreibt den Standardparameter *mailbox.default.connection.protocol*.

mailbox.1.connection.username

- **Modul**: cmas-nimh
- **Beschreibung**: Benutzername für das erste konfigurierte Postfach. Überschreibt den Standardparameter *mailbox.default.connection.username*.

mailbox.2.connection.host

- **Modul**: cmas-nimh
- **Beschreibung**: Host (Server) für das zweite konfigurierte Postfach. Überschreibt den Standardparameter *mailbox.default.connection.host*.

mailbox.2.connection.password

- **Modul**: cmas-nimh
- **Beschreibung**: Passwort für das zweite konfigurierte Postfach. Überschreibt den Standardparameter *mailbox.default.connection.password*.

mailbox.2.connection.port

- **Modul**: cmas-nimh
- **Beschreibung**: Port für das zweite konfigurierte Postfach. Überschreibt den Standardparameter *mailbox.default.connection.port*.

mailbox.2.connection.protocol

- **Modul**: cmas-nimh
- **Beschreibung**: Protokoll (z. B. IMAP oder POP3) für das zweite konfigurierte Postfach. Überschreibt den Standardparameter *mailbox.default.connection.protocol*.

mailbox.2.connection.username

- **Modul**: cmas-nimh
- **Beschreibung**: Benutzername für das zweite konfigurierte Postfach. Überschreibt den Standardparameter *mailbox.default.connection.username*.

Für alle Postfach-Properties im Zusammenhang mit NIMH gilt folgendes Prinzip: Es wird eine Standard-Property definiert (z.B *mailbox.default.connection.port*). Wenn keine postfachspezifische Property konfiguriert ist, wird dieser Standardwert verwendet.

mailbox.default.connection.host

- **Modul**: cmas-nimh
- **Beschreibung**: Host (Servername) eines bestimmten Postfachs, aus dem der Poller E-Mails liest.
- **Typ**: string
- **Neustart erforderlich**: nein
- **System**: nein

- **Optional**: ja
- **Beispielwert**: 10.10.1.157
- **Seit**: 6.4.0

mailbox.default.connection.password

- **Modul**: cmas-nimh
- **Beschreibung**: Passwort für ein bestimmtes Postfach, aus dem der Poller E-Mails liest.
- **Typ**: string
- **Neustart erforderlich**: nein
- **System**: nein
- **Optional**: ja
- **Beispielwert**: consol
- **Seit**: 6.4.0

mailbox.default.connection.port

- **Modul**: cmas-nimh
- **Beschreibung**: Port für ein bestimmtes Postfach, aus dem der Poller E-Mails liest.
- **Typ**: string
- **Neustart erforderlich**: nein
- **System**: nein
- **Optional**: ja
- **Beispielwert**: 143
- **Seit**: 6.4.0

mailbox.default.connection.protocol

- **Modul**: cmas-nimh
- **Beschreibung**: Protokoll des Pollers, z. B. IMAP oder POP3. Kein Standardwert.
- **Typ**: string
- **Neustart erforderlich**: nein
- **System**: nein
- **Optional**: ja
- **Beispielwert**: imap
- **Seit**: 6.4.0

mailbox.default.connection.username

- **Modul**: cmas-nimh
- **Beschreibung**: Benutzername für ein bestimmtes Postfach, aus dem der Poller E-Mails liest.

- **Typ**: string
- **Neustart erforderlich**: nein
- **System**: nein
- **Optional**: ja
- **Beispielwert**: username
- **Seit**: 6.4.0

mailbox.default.session.mail.debug

- **Modul**: cmas-nimh
- **Beschreibung**: Beispiel für javax.mail-Property - ermöglicht detaillierteres Session-Debugging mit javax.mail.
- **Typ**: boolean
- **Neustart erforderlich**: nein
- **System**: nein
- **Optional**: ja
- **Beispielwert**: true
- **Seit**: 6.4.0

mailbox.default.session.mail.imap.timeout

- **Modul**: cmas-nimh
- **Beschreibung**: Beispiel für javax.mail-Property
- **Typ**: integer
- **Neustart erforderlich**: nein
- **System**: nein
- **Optional**: ja
- **Beispielwert**: 120
- **Seit**: 6.4.0

mailbox.default.session.mail.mime.address.strict

- **Modul**: cmas-nimh
- **Beschreibung**: Beispiel für javax.mail-Property - Gegenstück zum alten *mail.mime.strict* von Mule, erlaubt das nicht so strikte Parsen des E-Mail-Headers.
- **Typ**: boolean
- **Neustart erforderlich**: nein
- **System**: nein
- **Optional**: ja

- **Beispielwert**: true
- **Seit**: 6.4.0

mailbox.default.session.mail.pop3.timeout

- **Modul**: cmas-nimh
- **Beschreibung**: Beispiel für javax.mail-Property
- **Typ**:
- **Neustart erforderlich**:
- **System**:
- **Optional**:
- **Beispielwert**:
- **Seit**: 6.4.0

mailbox.default.session.mail.<protocol>.partialfetch

- **Modul**: cmas-nimh
- **Beschreibung**: z. B. mailbox.default.session.mail.imaps.partialfetch
- **Typ**: boolean
- **Neustart erforderlich**: nein
- **System**: nein
- **Optional**: ja
- **Beispielwert**: false
- **Seit**:

mailbox.default.task.delete.read.messages

- **Modul**: cmas-nimh
- **Beschreibung**: Legt fest, ob Nachrichten nach dem Lesen aus dem Postfach entfernt werden sollen. Beim IMAP-Protokoll werden Nachrichten standardmäßig als SEEN gekennzeichnet. Bei POP3 wird die Nachricht nur gelöscht, wenn der Wert auf true gesetzt ist. Anderenfalls bleibt die Nachricht auf dem Server, was zu einer Endlos-Leseschleife führt. Standardwert: false.
- **Typ**: boolean
- **Neustart erforderlich**: nein
- **System**: nein
- **Optional**: ja
- **Beispielwert**: false
- **Seit**: 6.4.0

mailbox.default.task.enabled

- **Modul**: cmas-nimh
- **Beschreibung**: Mit dieser System-Property kann der Service Thread eines bestimmten Pollers deaktiviert werden. Standardwert: true.
- **Typ**: boolean
- **Neustart erforderlich**: nein
- **System**: nein
- **Optional**: ja
- **Beispielwert**: false
- **Seit**: 6.4.0

mailbox.default.task.interval.seconds

- **Modul**: cmas-nimh
- **Beschreibung**: Standard-Intervall für den Abruf von Postfächern. Standardwert: 60 Sekunden
- **Typ**: integer
- **Neustart erforderlich**: nein
- **System**: nein
- **Optional**: ja
- **Beispielwert**: 60
- **Seit**: 6.4.0

mailbox.default.task.max.message.size

- **Modul**: cmas-nimh
- **Beschreibung**: Maximale Größe von E-Mails (d. h. E-Mail plus Attachments). E-Mails, die größer als dieser Wert sind, werden nicht automatisch von NIMH verarbeitet. Sie werden in der Datenbank (Tabelle *cmas_nimh_archived_mail*) gespeichert und erscheinen daher in den E-Mail-Backups im Admin Tool (siehe Abschnitt *E-Mail-Backups* im *ConSol CM Administratorhandbuch*). Von dort können sie erneut gesendet, in das Dateisystem geladen oder gelöscht werden. Für diese Operationen ist die Größe der Nachricht nicht relevant. Standardmäßig auf 10 MB (10485760) gesetzt.
- **Typ**: integer
- **Neustart erforderlich**: nein
- **System**: nein
- **Optional**: ja
- **Beispielwert**: 10485760
- **Seit**: 6.4.0

mailbox.default.task.max.messages.per.run

- **Modul**: cmas-nimh
- **Beschreibung**: Anzahl der Nachrichten, die gleichzeitig aus dem Postfach abgeholt werden. Muss mit dem Transaktions-Timeout korrelieren. Standardwert: 20.
- **Typ**: integer
- **Neustart erforderlich**: nein
- **System**: nein
- **Optional**: ja
- **Beispielwert**: 60
- **Seit**: 6.4.0

mailbox.default.task.timeout.seconds

- **Modul**: cmas-nimh
- **Beschreibung**: Nach dieser Zeit (der Inaktivität) wird der Service Thread als beschädigt betrachtet und automatisch neu gestartet. Standardwert: 120 Sekunden
- **Typ**: integer
- **Neustart erforderlich**: nein
- **System**: nein
- **Optional**: ja
- **Beispielwert**: 60
- **Seit**: 6.4.0

mailbox.default.task.transaction.timeout.seconds

- **Modul**: cmas-nimh
- **Beschreibung**: Standard-Transaktions-Timeout für Transaktionen, die E-Mails abrufen. Sollte mit der Anzahl der Nachrichten, die gleichzeitig abgeholt werden, korrelieren. Standardwert: 60 Sekunden
- **Typ**: integer
- **Neustart erforderlich**: nein
- **System**: nein
- **Optional**: ja
- **Beispielwert**: 60
- **Seit**: 6.4.0

mailbox.polling.threads.mail.log.enabled

- **Modul**: cmas-nimh
- **Description**: Ermöglicht das Protokollieren von E-Mails, was in Cluster-Umgebungen besonders wichtig ist (wird dort als Semaphor genutzt).
- **Typ**: boolean
- **Neustart erforderlich**: nein
- **System**: nein
- **Optional**: ja
- **Beispielwert**: true (Standardwert)
- **Seit**: 6.9.4.1

mailbox.polling.threads.number

- **Modul**: cmas-nimh
- **Beschreibung**: Anzahl der Threads für den Zugriff auf Postfächer. Standardwert: 1.
- **Typ**: integer
- **Neustart erforderlich**: nein
- **System**: nein
- **Optional**: ja
- **Beispielwert**: 1
- **Seit**: 6.4.0

queue.polling.threads.number

- **Modul**: cmas-nimh
- **Beschreibung**: Anzahl der Threads, die zur Überwachung der E-Mail-Warteschlange gestartet werden. Standardwert: 1.
- **Typ**: integer
- **Neustart erforderlich**: nein
- **System**: nein
- **Optional**: ja
- **Beispielwert**: 1
- **Seit**: 6.4.0

queue.polling.threads.shutdown.timeout.seconds

- **Modul**: cmas-nimh
- **Beschreibung**: Wartezeit nach dem Shutdown-Signal. Wenn der Timeout erreicht ist, wird der Thread beendet. Standardwert: 60.
- **Typ**: integer

- **Neustart erforderlich**: nein
- **System**: nein
- **Optional**: ja
- **Beispielwert**: 60
- **Seit**: 6.4.0

queue.polling.threads.watchdog.interval.seconds

- **Modul**: cmas-nimh
- **Beschreibung**: Intervall des Watchdog Threads. Standardwert: 30.
- **Typ**: integer
- **Neustart erforderlich**: nein
- **System**: nein
- **Optional**: ja
- **Beispielwert**: 30
- **Seit**: 6.4.0

queue.task.error.pause.seconds

- **Modul**: cmas-nimh
- **Beschreibung**: Anzahl der Sekunden, die der Queue-Poller nach einem Infrastrukturfehler (z. B. der Datenbank) wartet. Standardwert: 180 Sekunden
- **Typ**: integer
- **Neustart erforderlich**: nein
- **System**: nein
- **Optional**: ja
- **Beispielwert**: 180
- **Seit**: 6.4.0

queue.task.interval.seconds

- **Modul**: cmas-nimh
- **Beschreibung**: Thread-Intervall zur Überwachung der Haupt-E-Mail-Warteschlange. Standardwert: 15.
- **Typ**: integer
- **Neustart erforderlich**: nein
- **System**: nein
- **Optional**: ja
- **Beispielwert**: 15
- **Seit**: 6.4.0

queue.task.max.retries

- **Modul**: cmas-nimh
- **Beschreibung**: Maximale Anzahl der Versuche, eine E-Mail nach einer Exception erneut zu verarbeiten. Ist diese erreicht, wird die E-Mail archiviert. Die archivierte E-Mail kann über das NIMH API (oder das Admin Tool) wieder aktiviert werden.
- **Typ**: integer
- **Neustart erforderlich**: nein
- **System**: nein
- **Optional**: ja
- **Beispielwert**: 10
- **Seit**: 6.4.0

queue.task.timeout.seconds

- **Modul**: cmas-nimh
- **Beschreibung**: Nach dieser Zeit (der Inaktivität) wird der Service Thread als beschädigt betrachtet und automatisch neu gestartet. Standardwert: 600 Sekunden
- **Typ**: integer
- **Neustart erforderlich**: nein
- **System**: nein
- **Optional**: ja
- **Beispielwert**: 600
- **Seit**: 6.4.0

queue.task.transaction.timeout.seconds

- **Modul**: cmas-nimh
- **Beschreibung**: Transaktions-Timeout für die E-Mail-Verarbeitung in der Warteschlange. Standardwert: 60.
- **Typ**: integer
- **Neustart erforderlich**: nein
- **System**: nein
- **Optional**: ja
- **Beispielwert**: 60
- **Seit**: 6.4.0

G.2.2.11 cmas-nimh-extension (Modul)

mail.attachments.validation.info.sender

- **Modul**: cmas-nimh-extension
- **Beschreibung**: Setzt den From-Header bei Attachments vom Typ *error notification mail*.
- **Typ**: string
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: admin@mail.com
- **Seit**: 6.7.5

> Diese System-Property entspricht der alten *cmas-esb-mail, mail.attachments.validation.info.sender*

mail.attachments.validation.info.subject

- **Modul**: cmas-nimh-extension
- **Beschreibung**: Setzt den Betreff bei Attachments vom Typ error notification mail.
- **Typ**: string
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: E-Mail konnte nicht verarbeitet werden, weil ihre Attachments zurückgewiesen wurden!
- **Seit**: 6.7.5

> Diese System-Property entspricht der alten *cmas-esb-mail, mail.attachments.validation.info.subject*

mail.db.archive

- **Modul**: cmas-nimh-extension
- **Beschreibung**: Wenn dieser Wert auf *true* gesetzt ist, werden eingehende E-Mails in der Datenbank archiviert.
- **Typ**: boolean
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: ja

- **Beispielwert**: false (Standardwert)
- **Seit**: 6.8.5.5

mail.error.from.address

Diese System-Property entspricht der alten *cmas-esb-mail,mail.mule.service*

mail.error.to.address

Diese System-Property entspricht der alten *cmas-esb-mail, mail.process.error*

mail.on.error

- **Modul**: cmas-nimh-extension
- **Beschreibung**: Wenn diese Property auf *true* gesetzt ist, wird im Fall, dass eine E-Mail nicht verarbeitet werden konnte, ein Fehler-E-Mail an die oben konfigurierten E-Mail-Adressen gesendet. Standardwert: true.
- **Typ**: boolean
- **Neustart erforderlich**: nein
- **System**: nein
- **Optional**: ja
- **Beispielwert**: false
- **Seit**: 6.4.0

mail.process.error

- **Modul**: cmas-nimh-extension
- **Beschreibung**: To-Adresse für E-Mails mit Fehlermeldungen des Mule.
- **Typ**: email
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: myuser@consol.de
- **Seit**: 6.4.0

mail.ticketname.pattern

- **Modul**: cmas-nimh-extension
- **Beschreibung**: Regulärer Ausdruck, mit dem der Ticketname im Betreff eingehender E-Mails identifiziert wird.
- **Typ**: string

- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: .*?Ticket\s+\((\S+)\).*
- **Seit**: 6.4.0

G.2.2.12 cmas-restapi-core (Modul)

security.fields.customer.exposure.check.enabled

- **Modul**: cmas-restapi-core
- **Beschreibung**: Aktiviert die Prüfung der Annotation *customer exposure* für Benutzerdefinierte Felder.
- **Typ**: boolean
- **Neustart erforderlich**: nein
- **System**: nein
- **Optional**: ja
- **Beispielwert**: true (Standardwert)
- **Seit**: 6.10.5.4

G.2.2.13 cmas-setup-hibernate (Modul)

cmas.dropSchemaBeforeSetup

- **Modul**: cmas-setup-hibernate
- **Beschreibung**: Gibt an, ob das Schema während des Setups gelöscht werden soll (wurde).
- **Typ**: string
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: true
- **Seit**: 6.0

hibernate.dialect

- **Modul**: cmas-setup-hibernate
- **Beschreibung**: Der Hibernate-Dialekt. Normalerweise wird dieser Wert während des initialen Setups gesetzt (abhängig vom Datenbanksystem).
- **Typ**: string
- **Neustart erforderlich**: nein
- **System**: ja

- **Optional**: nein
- **Beispielwert**: org.hibernate.dialect.MySQL5InnoDBDialect
- **Seit**: 6.0

G.2.2.14 cmas-setup-manager (Modul)

initialized

- **Modul**: cmas-setup-manager
- **Beschreibung**: Legt fest, ob CMAS initialisiert ist. Wenn dieser Wert fehlt oder nicht *auf* true gesetzt ist, wird das Setup ausgeführt.
- **Typ**: boolean
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: true
- **Seit**: 6.0

> ⚠ Seien Sie mit der Verwendung dieser System-Property sehr vorsichtig! Wenn Sie den Wert auf *false* setzen, wird der ConSol CM-Server beim nächsten Systemstart das System-Setup ausführen, d. h. alle Daten des bestehenden Systems gehen verloren, einschließlich der System-Properties!

G.2.2.15 cmas-setup-scene (Modul)

scene

- **Modul**: cmas-setup-scene
- **Beschreibung**: Szenariodatei, die während des Setups importiert wurde (kann leer gelassen werden).
- **Typ**: string
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: vfszip:/P:/dist/target/jboss/server/cmas/deploy/cm-dist-6.5.1-SNAPSHOT.ear/APP-INF/lib/dist-scene-6.5.1-SNAPSHOT.jar/META-INF/cmas/scenes/helpdesk-sales_scene.jar/
- **Seit**: 6.0

G.2.2.16 cmas-workflow-engine (Modul)

jobExecutor.adminMail

- **Modul**: cmas-workflow-engine
- **Beschreibung**: E-Mail-Adresse, an die Benachrichtigungs-E-Mails, die Probleme der Job-ausführung betreffen, geschickt werden (wenn die Anzahl der Neuversuche überschritten wurde).
- **Typ**: string
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: ja
- **Beispielwert**: admin@consol.de
- **Seit**: 6.8.0

jobExecutor.idleInterval.seconds

- **Modul**: cmas-workflow-engine
- **Beschreibung**: Legt fest, wie oft der Job Executor Thread nach neuen Jobs zum Ausführen sucht.
- **Typ**: integer
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: ja
- **Beispielwert**: 45 (Standardwert bis CM-Version 6.10.5.2. Der Standardwert für CM-Versionen 6.10.5.3 und höher ist 5)
- **Seit**: 6.8.0

jobExecutor.jobMaxRetries

- **Modul**: cmas-workflow-engine
- **Beschreibung**: Steuert die Anzahl der erneuten Versuche, die der Job Executor unternimmt, bevor er einen Job als fehlgeschlagen deklariert.
- **Typ**: integer
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: ja
- **Beispielwert**: 5 (Standardwert)
- **Seit**: 6.8.0

jobExecutor.jobMaxRetriesReachedSubject

- **Modul**: cmas-workflow-engine
- **Beschreibung**: Der Betreff für Benachrichtigungs-E-Mails, die Administratoren über fehlgeschlagene Job-Ausführungen erhalten.
- **Typ**: string
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: ja
- **Beispielwert**: Maximale Anzahl der Neuversuche für Job erreicht. Job wurde entfernt! (Standard)
- **Seit**: 6.8.0

jobExecutor.lockingLimit

- **Modul**: cmas-workflow-engine
- **Beschreibung**: Anzahl der gleichzeitig gelockten (als "in der Ausführung" markierten) Jobs des Job Executor Threads.
- **Typ**: integer
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: ja
- **Beispielwert**: 5 (Standardwert seit CM-Version 6.10.5.3)
- **Seit**: 6.8.0

jobExecutor.lockTimeout.seconds

- **Modul**: cmas-workflow-engine
- **Beschreibung**: Legt fest, wie lange ein Job vom Job Executor (als "in der Ausführung" markiert) gelockt werden kann.
- **Typ**: integer
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: ja
- **Beispielwert**: 360 (Standardwert)
- **Seit**: 6.8.0

jobExecutor.mailFrom

- **Modul**: cmas-workflow-engine
- **Beschreibung**: E-Mail-Adresse, die als From-Header für Admin-Benachrichtigungen eingesetzt wird.
- **Typ**: string
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: ja
- **Beispielwert**: jobexecutor@consol.de
- **Seit**: 6.8.0

jobExecutor.maxInactivityInterval.minutes

- **Modul**: cmas-workflow-engine
- **Beschreibung**: Länge der erlaubten Inaktivität des Job Executors in Minuten (z. B. wenn er durch eine Langzeit-Ausführung gesperrt wird). Nach dieser Zeit werden die Executor Threads neu gestartet.
- **Typ**: integer
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: ja Standardwert ist auf 30 Minuten gesetzt.
- **Beispielwert**: 15 (Standardwert)
- **Seit**: 6.9.2.0

jobExecutor.threads

- **Modul**: cmas-workflow-engine
- **Beschreibung**: Anzahl der Job Executor Threads.
- **Typ**: integer
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: ja
- **Beispielwert**: 1 (Standardwert)
- **Seit**: 6.8.0

jobExecutor.timerRetryInterval.seconds

- **Modul**: cmas-workflow-engine
- **Beschreibung**: Legt fest, wie lange der Job Executor Thread nach einem Fehler bei der Job-Ausführung wartet.

- **Typ**: integer
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: ja
- **Beispielwert**: 10 (Standardwert bis CM-Version 6.10.5.2. Der Standardwert für CM-Versionen 6.10.5.3 und höher ist 30.)
- **Seit**: 6.8.0

jobExecutor.txTimeout.seconds

- **Modul**: cmas-workflow-engine
- **Beschreibung**: Transaktions-Timeout für die Job-Ausführung.
- **Typ**: integer
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: ja
- **Beispielwert**: 60 (Standardwert)
- **Seit**: 6.8.0

G.2.2.17 cmas-workflow-jbpm (Modul)

fetchLock.interval

- **Modul**: cmas-workflow-jbpm
- **Beschreibung**:
- **Typ**: integer
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: 5000
- **Entfernt in**: 6.8.0

jobExecutor.idleInterval

- **Modul**: cmas-workflow-jbpm
- **Beschreibung**:
- **Typ**: integer
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein

- **Beispielwert**: 45000
- **Entfernt in**: 6.8.0
- **Ersetzt durch**: jobExecutor.idleInterval.seconds

jobExecutor.jobExecuteRetryNumber

- **Modul**: cmas-workflow-jbpm
- **Beschreibung**:
- **Typ**: integer
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: 5
- **Entfernt in**: 6.8.0
- **Ersetzt durch**: jobExecutor.jobMaxRetries

jobExecutor.timerRetryInterval

- **Modul**: cmas-workflow-jbpm
- **Beschreibung**:
- **Typ**: integer
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: 10000
- **Entfernt in**: 6.8.0
- **Ersetzt durch**: jobExecutor.timerRetryInterval.seconds

mail.sender.address

- **Modul**: cmas-workflow-jbpm
- **Beschreibung**: From-Adresse für E-Mails, die von der Workflow-Engine versendet werden.
- **Typ**: string
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: myuser@consol.de
- **Entfernt in**: 6.8.0
- **Ersetzt durch**: jobExecutor.mailFrom

outdated.lock.age

- **Modul**: cmas-workflow-jbpm
- **Beschreibung**:
- **Typ**: integer
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: 60000
- **Entfernt in**: 6.8.0
- **Ersetzt durch**: cmas-workflow-engine, jobExecutor.lockTimeout.seconds

refreshTimeInCaseOfConcurrentRememberMeRequests

- **Modul**: cmas-workflow-jbpm
- **Beschreibung**: Legt die Aktualisierungszeit (in Sekunden) fest, nach der die Seite im Falle von gleichzeitigen Anfragen von Angemeldet bleiben neu geladen wird. Diese Funktion verhindert, dass ein Benutzer zu viele Lizenzen in Anspruch nimmt. Erhöhen Sie die Zeit, wenn immer noch Sessions belegt werden.
- **Typ**: integer
- **Neustart erforderlich**: ja
- **System**: ja
- **Optional**: ja
- **Beispielwert**: 5
- **Seit**: 6.8.2

G.2.2.18 cmweb-server-adapter (Modul)

attachment.upload.timeout

- **Modul**: cmweb-server-adapter
- **Beschreibung**: Definiert das Transaktions-Timeout in Minuten für das Hinzufügen von Attachments zu einem Ticket, einer Ressource oder einem Kunden. Dabei zählt die Zeit für den Upload aller Attachments einer Transaktion. Wenn das Timeout eintritt, werden alle Dateien, die temporär auf dem Server gespeichert waren, gelöscht. Es wird keine Datei hochgeladen.
- **Typ**: Integer
- **Neustart erforderliche**: nein
- **System**: ja
- **Optional**: ja
- **Beispielwert**: 3
- **Seit**: 6.10.5.3

automatic.booking.enabled

- **Modul**: cmweb-server-adapter
- **Description**: Wenn aktiviert, wird die Zeit für das Erstellen eines Kommentars oder einer E-Mail gemessen und eine automatische Zeitbuchung hinzugefügt.
- **Typ**: boolean
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: ja
- **Beispielwert**: true
- **Seit**: 6.9.4.2

checkUserOnlineIntervalInSeconds

- **Modul**: cmweb-server-adapter
- **Beschreibung**: Das Intervall (in Sekunden), in dem geprüft wird, welche Benutzer online sind (Standard 180 Sekunden = 3 Minuten).
- **Typ**: integer
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: 180
- **Seit**: 6.0
- **Entfernt in**: 6.5 / 6.11.0.1

cmoffice.enabled

- **Modul**: cmweb-server-adapter
- **Beschreibung**: Legt fest, ob CM.Doc (vorher CM/Office) aktiviert ist.
- **Typ**: boolean
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: false
- **Seit**: 6.4.0

cmoffice.strict.versioning.enabled

- **Modul**: cmweb-server-adapter
- **Beschreibung**: Steuert, ob der Speichervorgang für Microsoft Word- / OpenOffice-Dokumente ein neues Attachment erzeugt (*true*) oder das vorhandene Attachment überschreibt (*false*). Dies betrifft das Verhalten innerhalb einer Session mit dem Textbearbeitungsprogramm. Wenn das Programm beendet wird, funktioniert der Mechanismus zum Überschreiben nicht mehr.
- **Typ**: Boolean
- **Neustart erforderlich**: nein
- **System**: nein
- **Optional**: ja
- **Beispielwert**: true
- **Seit**: 6.10.5.4

commentRequiredForTicketCreation

- **Modul**: cmweb-server-adapter
- **Beschreibung**: Legt fest, ob der Kommentar ein Pflichtfeld für die Erstellung eines Tickets ist.
- **Typ**: boolean
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: true (Standardwert)
- **Seit**: 6.2.0

customizationVersion

- **Modul**: cmweb-server-adapter
- **Beschreibung**: UID der letzten Version der Web-Anpassung. Wird nur intern verwendet. Der Wert darf nicht geändert werden.
- **Typ**: string
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: cd58453e-f3cc-4538-8030-d15e8796a4a7
- **Seit**: 6.5.0

data.optimization

- **Modul**: cmweb-server-adapter
- **Beschreibung**: Definiert die Optimierung der Response-Daten. Bisher werden die folgenden Werte unterstützt (mehrere Werte können durch '|' getrennt gesetzt werden): MINIFICATION und COMPRESSION. MINIFICATION minimiert HTML-Daten, indem z. B. Leerzeichen und Kommentare entfernt werden. COMPRESSION wendet gzip-Komprimierung auf die HTTP-Response an. (Hinweis: Wenn das System im Cluster-Modus läuft und Sie parallel verschiedene Konfigurationen testen möchten, können Sie für jeden Cluster-Node verschiedene Werte setzen, indem Sie die System-Property nach dem Muster *data.optimization.nodeId* spezifizieren, um die Standard-Property zu überschreiben.)
- **Typ**: string
- **Neustart**: COMPRESSION kann ohne Neustart ein- und ausgeschaltet werden, für MINIFICATION ist ein Neustart erforderlich.
- **System**: ja
- **Optional**: ja
- **Beispielwert**: MINIFICATION|COMPRESSION

defaultContentEntryClassName

- **Modul**: cmweb-server-adapter
- **Beschreibung**: Standard-Textklasse für neue ACIMs.
- **Typ**: string
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: default_class
- **Seit**: 6.3.0

defaultNumberOfCustomFieldsColumns

- **Modul**: cmweb-server-adapter
- **Beschreibung**: Standardanzahl an Spalten für Benutzerdefinierte Felder.
- **Typ**: integer
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: 3
- **Seit**: 6.2.0

diffTrackingEnabled

- **Modul**: cmweb-server-adapter
- **Beschreibung**: Legt fest, ob das gleichzeitige Bearbeiten eines Tickets durch verschiedene Bearbeiter möglich ist. Der Standardwert ist *true*.
  false: Vorherige Art und Weise der Behandlung von Änderungen beim Editieren von Tickets. Wenn ein Ticket zwischendurch geändert wurde, kann der aktuelle Bearbeiter seine Änderungen nicht speichern, ohne die Seite vorher neu zu laden.
  true: Neuer Modus zur Behandlung von Änderungen. Wenn das Ticket geändert wurde, wird das Speichern von anderen Änderungen nicht mehr blockiert. Wenn der Teil des Tickets, der geändert wurde, genau der Teil ist, der vom speichernden Bearbeiter geändert wird, wird eine Informationsmeldung angezeigt, die Ticketänderungen werden aber trotzdem gespeichert.
- **Typ**: boolean
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: true
- **Seit**: 6.10.1
- **Entfernt in**: 6.11.0

favoritesSizeLimit

- **Modul**: cmweb-server-adapter
- **Beschreibung**: Maximale Anzahl von Elementen in der Favoritenliste.
- **Typ**: integer
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: 10
- **Seit**: 6.0

globalSearchResultSizeLimit

- **Modul**: cmweb-server-adapter
- **Beschreibung**: Maximale Anzahl an Ergebnissen in der Schnellsuche.
- **Typ**: integer
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: 10
- **Seit**: 6.0

helpFilePath

- **Modul**: cmweb-server-adapter
- **Beschreibung**: URL für die Onlinehilfe. Wenn der Wert nicht leer gelassen wird, wird der Hilfe-Button im Web Client angezeigt.
- **Typ**: string
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: ja
- **Beispielwert**: http://www.consol.de
- **Seit**: 6.2.1

hideTicketSubject

- **Modul**: cmweb-server-adapter
- **Beschreibung**: Wenn auf *true* gesetzt, wird das Ticketthema ausgeblendet.
- **Typ**: boolean
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: false
- **Seit**: 6.2.1

mail.from

- **Modul**: cmweb-server-adapter
- **Beschreibung**: Diese E-Mail-Adresse wird anstelle der E-Mail-Adresse des Bearbeiters in E-Mail-Konversationen verwendet.
- **Typ**: string
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: ja
- **Seit**: 6.1.2

mail.reply.to

- **Modul**: cmweb-server-adapter
- **Beschreibung**: Wenn dieser Wert gesetzt ist, zeigt der Web Client diesen Wert beim Versenden einer E-Mail im Reply-To-Feld an.
- **Typ**: string
- **Neustart erforderlich**: nein

- **System**: ja
- **Optional**: ja
- **Seit**: 6.0.1

> ⚠ Bitte lesen Sie dazu die ausführlichen Informationen über Reply-To-Adressen in ConSol CM im Abschnitt *Skripte des Typs E-Mail* im *ConSol CM Administratorhandbuch*.

mailTemplateAboveQuotedText

- **Modul**: cmweb-server-adapter
- **Beschreibung**: Legt das Verhalten der E-Mail-Vorlagen im Ticket-E-Mail-Editor fest, wenn eine andere E-Mail zitiert wird, d. h. auf diese geantwortet oder diese weitergeleitet wird. Wird oft verwendet, um die Signatur korrekt zu platzieren.
- **Typ**: boolean
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: false
- **Seit**: 6.2.4

maxSizePerPagemapInMegaBytes

- **Modul**: cmweb-server-adapter
- **Beschreibung**: Maximale Größe (in MB) für jede Wicket Pagemap.
- **Typ**: integer
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: 15
- **Seit**: 6.3.5

pagemapLockDurationInSeconds

- **Modul**: cmweb-server-adapter
- **Beschreibung**: Anzahl der Sekunden, die vergehen müssen, bevor eine Pagemap als zu lange gelockt angesehen wird.
- **Typ**: integer
- **Neustart erforderlich**: ja
- **System**: ja

- **Optional**: ja
- **Beispielwert**: 60
- **Seit**: 6.7.3

postActivityExecutionScriptName

- **Modul**: cmweb-server-adapter
- **Beschreibung**: Definiert den Namen des Skripts, das nach jeder Workflow-Aktivität ausgeführt wird (siehe Abschnitt *PostActivityExecutionScript* im *ConSol CM Administratorhandbuch*. Wenn kein Skript ausgeführt werden soll, lassen Sie diesen Wert leer.
- **Typ**: string
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: postActivityExecutionHandler
- **Seit**: 6.2.0

queuesExcludedFromGS

- **Modul**: cmweb-server-adapter
- **Beschreibung**: Durch Kommas getrennte Liste von Queue-Namen, die von der Schnellsuche ausgeschlossen werden sollen.
- **Typ**: string
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: ja
- **Seit**: 6.0

rememberMeLifetimeInMinutes

- **Modul**: cmweb-server-adapter
- **Beschreibung**: Lebensdauer für *Angemeldet bleiben* in Minuten.
- **Typ**: integer
- **Neustart erforderlich**: ja
- **System**: ja
- **Optional**: nein
- **Beispielwert**: 1440
- **Seit**: 6.0

request.scope.transaction

- **Modul**: cmweb-server-adapter
- **Beschreibung**: Ermöglicht die Deaktivierung von Scope-Transaktionen für Requests. Standardmäßig wird pro Request eine Transaktion verwendet. Wenn Sie diese Property auf *false* setzen, wird eine Transaktion pro Aufruf einer Service-Methode erzeugt.
- **Typ**: boolean
- **Neustart erforderlich**: ja
- **System**: ja
- **Optional**: ja
- **Beispielwert**: true
- **Seit**: 6.8.1

searchPageSize

- **Modul**: cmweb-server-adapter
- **Beschreibung**: Standardgröße der Seiten für Suchergebnisse.
- **Typ**: integer
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: 20
- **Seit**: 6.0

searchPageSizeOptions

- **Modul**: cmweb-server-adapter
- **Beschreibung**: Optionen für die Größe der Seiten für Suchergebnisse.
- **Typ**: string
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: 10|20|30|40|50|75|100
- **Seit**: 6.0

serverPoolingInterval

- **Modul**: cmweb-server-adapter
- **Beschreibung**: Definiert die Zeit in Sekunden, nach der der Pooling-Server die Caches auf dem Web-Layer ungültig macht.
- **Typ**: integer

- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: 5
- **Seit**: 6.1.0

supportEmail

- **Modul**: cmweb-server-adapter
- **Beschreibung**:
- **Typ**: string
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: ja
- **Seit**: 6.0
- **Entfernt in**: 6.11.0.1

themeOverlay

- **Modul**: cmweb-server-adapter
- **Beschreibung**: Name des verwendeten Themen-Overlays.
- **Typ**: string
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: ja
- **Beispielwert**: consolINT
- **Seit**: 6.0

ticketListRefreshIntervalInSeconds

- **Modul**: cmweb-server-adapter
- **Beschreibung**: Aktualisierungsintervall für die Ticketliste (in Sekunden).
- **Typ**: integer
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: 180
- **Seit**: 6.0

ticketListSizeLimit

- **Modul**: cmweb-server-adapter
- **Beschreibung**: Maximale Anzahl von Tickets in der Ticketliste.
- **Typ**: integer
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: 100
- **Seit**: 6.0

unitIndexSearchResultSizeLimit

- **Modul**: cmweb-server-adapter
- **Beschreibung**: Maximale Anzahl der Units in der Ergebnisliste, wenn nach Units gesucht wird (Beispiel: Kontaktsuche).
- **Typ**: integer
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: 5
- **Seit**: 6.0

urlLogoutPath

- **Modul**: cmweb-server-adapter
- **Beschreibung**: URL die verwendet wird, wenn sich der Bearbeiter abmeldet. (wenn kein Wert gesetzt wird, wird nach dem Abmelden die Anmeldeseite angezeigt.)
- **Typ**: string
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: ja
- **Beispielwert**: http://intranet.consol.de
- **Seit**: 6.3.1

webSessionTimeoutInMinutes

- **Modul**: cmweb-server-adapter
- **Beschreibung**: Session-Timeout in Minuten.
- **Typ**: integer
- **Neustart erforderlich**: ja

- **System**: ja
- **Optional**: nein
- **Beispielwert**: 180
- **Entfernt in**: 6.7.1
- **Ersetzt durch**: cmas-core-server, server.session.timeout

wicketAjaxRequestHeaderFilterEnabled

- **Modul**: cmweb-server-adapter
- **Beschreibung**: Aktiviert Filter für Wicket AJAX-Anfragen, die auf veralteten Seiten mit Wicket 1.4-Skripten (CM-Version vor 6.8.0) stammen, nach einer Aktualisierung auf CM-Versionen ab 6.8.0.
- **Typ**: boolean
- **Neustart erforderlich**: ja
- **System**: ja
- **Optional**: ja
- **Beispielwert**: false
- **Seit**: 6.8.1

G.2.3 Liste der System-Properties nach Bereich

Dieses Kapitel enthält eine Liste der System-Properties, die für die folgenden Bereiche relevant sind:

- CMRF- und DWH-Konfiguration
- Konfiguration von Indexer und Suche
- LDAP-Konfiguration
- E-Mail-Konfiguration
- Konfiguration des Aktivitätsintervalls
- Administrator-E-Mail-Adressen

G.2.3.1 CMRF- und DWH-Konfiguration

autocommit.cf.changes

- **Modul**: cmas-dwh-server
- **Beschreibung**: Definiert, ob DWH-Aufgaben, die aufgrund von Konfigurationsänderungen an Benutzerdefinierten Feldern anfallen, automatisch ohne manuelle Interaktion im Admin Tool ausgeführt werden. Diese Property kann auch im Admin Tool im Navigationselement *DWH* gesetzt werden. Der Standardwert und empfohlene Wert ist *false*.
- **Typ**: boolean
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: false
- **Seit**: 6.7.0

batch-commit-interval

- **Modul**: cmas-dwh-server
- **Beschreibung**: Anzahl der Objekte in einer JMS-Nachricht. Höhere Werte bedeuten eine bessere Übertragungsperformance und größeren Speicherverbrauch.
- **Typ**: integer
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: ja
- **Beispielwert**: 100
- **Seit**: 6.0.0

communication.channel

- **Modul**: cmas-dwh-server
- **Beschreibung**: Kommunikationskanal, mögliche Werte sind DIRECT (Datenbank-Kommunikationskanal, Standardwert seit 6.9.4.1) oder JMS (Standardwert vor 6.9.4.1). Diese System-Property muss vor 6.9.4.1 extra hinzugefügt werden.
- **Typ**: string
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: DIRECT
- **Seit**: 6.8.5.0
- **Entfernt in**: 6.11.0.0

dwh.mode

- **Modul**: cmas-dwh-server
- **Beschreibung**: Aktueller Modus der DWH-Datenübermittlung. Mögliche Werte sind OFF, ADMIN, LIVE.
- **Typ**: string
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: OFF
- **Seit**: 6.0.1

ignore-queues

- **Modul**: cmas-dwh-server
- **Beschreibung**: Durch eine durch Kommas getrennte Liste von Queue-Namen wird hier festgelegt, dass Tickets dieser Queues nicht ins DWH übermittelt werden.
- **Typ**: string
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: ja
- **Beispielwert**: QueueName1,QueueName2,QueueName3
- **Seit**: 6.6.19
- **Entfernt in**: 6.8.1

is.cmrf.alive

- **Modul**: cmas-dwh-server
- **Beschreibung**: Als Startpunkt sollte die Zeit genommen werden, bei der zuletzt eine Nachricht an das CMRF gesendet wurde. Wenn nach diesem Wert (in Sekunden) keine Antwort vom CMRF empfangen wird, wird ein DWH-Betriebsstatus mit der Fehlermeldung, dass das CMRF nicht erreichbar ist, erstellt.
- **Typ**: integer
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: 1200
- **Seit**: 6.7.0

java.naming.factory.initial

- **Modul**: cmas-dwh-server
- **Beschreibung**: Factory-Java-Klasse für DWH context factory.
- **Typ**: string
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: org.jnp.interfaces.NamingContextFactory
- **Seit**: 6.0.1
- **Entfernt in**: 6.11.0.0

java.naming.factory.url.pkgs

- **Modul**: cmas-dwh-server
- **Beschreibung**:
- **Typ**: string
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: org.jboss.naming:org.jnp.interfaces
- **Seit**: 6.0.1
- **Entfernt in**: 6.11.0.0

java.naming.provider.url

- **Modul**: cmas-dwh-server
- **Beschreibung**: URL des Naming Provider.
- **Typ**: string
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: localhost
- **Seit**: 6.0.1
- **Entfernt in**: 6.11.0.0

notification.error.description

- **Modul**: cmas-dwh-server
- **Beschreibung**: Text für E-Mails mit Fehlermeldungen des DWH.

- **Typ**: string
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: Fehler aufgetreten
- **Seit**: 6.0.1

notification.error.from

- **Modul**: cmas-dwh-server
- **Beschreibung**: From-Adresse für E-Mails mit Fehlermeldungen des DWH.
- **Typ**: string
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: ja
- **Seit**: 6.0.1

notification.error.subject

- **Modul**: cmas-dwh-server
- **Beschreibung**: Betreff für E-Mails mit Fehlermeldungen des DWH.
- **Typ**: string
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: Fehler aufgetreten
- **Seit**: 6.0.1

notification.error.to

- **Modul**: cmas-dwh-server
- **Beschreibung**: To-Adresse für E-Mails mit Fehlermeldungen des DWH.
- **Typ**: string
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: myuser@consol.de
- **Seit**: 6.0.1

notification.finished_successfully.description

- **Modul**: cmas-dwh-server
- **Beschreibung**: Text für E-Mails des DWHs, wenn eine Übertragung erfolgreich beendet wurde.
- **Typ**: string
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: Übertragung erfolgreich beendet
- **Seit**: 6.0.1

notification.finished_successfully.from

- **Modul**: cmas-dwh-server
- **Beschreibung**: From-Adresse für E-Mails des DWHs, wenn eine Übertragung erfolgreich beendet wurde.
- **Typ**: string
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: ja
- **Seit**: 6.0.1

notification.finished_successfully.subject

- **Modul**: cmas-dwh-server
- **Beschreibung**: Betreff für E-Mails des DWHs, wenn eine Übertragung erfolgreich beendet wurde.
- **Typ**: string
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: Übertragung erfolgreich beendet
- **Seit**: 6.0.1

notification.finished_successfully.to

- **Modul**: cmas-dwh-server
- **Beschreibung**: To-Adresse für E-Mails des DWHs, wenn eine Übertragung erfolgreich beendet wurde.
- **Typ**: string
- **Neustart erforderlich**: ja

- **System**: ja
- **Optional**: nein
- **Beispielwert**: myuser@consol.de
- **Seit**: 6.0.1

notification.finished_unsuccessfully.description

- **Modul**: cmas-dwh-server
- **Beschreibung**: Text für E-Mails des DWHs, wenn eine Übertragung nicht erfolgreich beendet wurde.
- **Typ**: string
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: Übertragung nicht erfolgreich beendet
- **Seit**: 6.0.1

notification.finished_unsuccessfully.from

- **Modul**: cmas-dwh-server
- **Beschreibung**: From-Adresse für E-Mails des DWHs, wenn eine Übertragung nicht erfolgreich beendet wurde.
- **Typ**: string
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: ja
- **Seit**: 6.0.1

notification.finished_unsuccessfully.subject

- **Modul**: cmas-dwh-server
- **Beschreibung**: Betreff für E-Mails des DWHs, wenn eine Übertragung nicht erfolgreich beendet wurde.
- **Typ**: string
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: Übertragung nicht erfolgreich beendet
- **Seit**: 6.0.1

notification.finished_unsuccessfully.to

- **Modul**: cmas-dwh-server
- **Beschreibung**: To-Adresse für E-Mails des DWHs, wenn eine Übertragung nicht erfolgreich beendet wurde.
- **Typ**: string
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: myuser@consol.de
- **Seit**: 6.0.1

notification.host

- **Modul**: cmas-dwh-server
- **Description**: Hostname des E-Mail-Servers (SMTP) für das Senden von DWH-E-Mails.
- **Typ**: string
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: ja
- **Beispielwert**: myserver.consol.de
- **Seit**: 6.0.1

notification.password

- **Modul**: cmas-dwh-server
- **Beschreibung**: Passwort für das Senden von DWH-E-Mails. (optional)
- **Typ**: string
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: ja
- **Seit**: 6.0.1

notification.port

- **Modul**: cmas-dwh-server
- **Beschreibung**: SMTP-Port für das Senden von DWH-E-Mails.
- **Typ**: string
- **Neustart erforderlich**: nein
- **System**: ja

- **Optional**: ja
- **Beispielwert**: 25
- **Seit**: 6.0.1

notification.protocol

- **Modul**: cmas-dwh-server
- **Beschreibung**: Das Protokoll, das für das Senden von E-Mails aus dem DWH verwendet wird.
- **Typ**: string
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: ja
- **Beispielwert**: pop3\

notification.username

- **Modul**: cmas-dwh-server
- **Beschreibung**: Benutzername (SMTP) für das Senden von DWH-E-Mails.
- **Typ**: string
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: ja
- **Beispielwert**: myuser
- **Seit**: 6.0.1

skip-ticket

- **Modul**: cmas-dwh-server
- **Beschreibung**: Tickets werden bei der Übertragung/Aktualisierung nicht übermittelt.
- **Typ**: boolean
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: false
- **Seit**: 6.6.19
- **Entfernt in**: 6.8.1

skip-ticket-history

- **Modul**: cmas-dwh-server
- **Beschreibung**: Ticketprotokoll wird bei der Übertragung/Aktualisierung nicht übermittelt.

- **Typ**: boolean
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: false
- **Seit**: 6.6.19
- **Entfernt in**: 6.8.1

skip-unit

- **Modul**: cmas-dwh-server
- **Beschreibung**: Units werden bei der Übertragung/Aktualisierung nicht übermittelt.
- **Typ**: boolean
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: false
- **Seit**: 6.6.19
- **Entfernt in**: 6.8.1

skip-unit-history

- **Modul**: cmas-dwh-server
- **Beschreibung**: Unit-Protokoll wird bei der Übertragung/Aktualisierung nicht übermittelt.
- **Typ**: boolean
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: false
- **Seit**: 6.6.19
- **Entfernt in**: 6.8.1

split.history

- **Modul**: cmas-dwh-server
- **Beschreibung**: Ändert das SQL-Statement dahingehend, dass Ticketprotokolle während der Übertragung an das DWH nicht für alle Tickets auf einmal abgeholt werden, sondern für ein Ticket pro SQL-Statement.
- **Typ**: boolean
- **Neustart erforderlich**: nein

- **System**: ja
- **Optional**: ja
- **Beispielwert**: false
- **Seit**: 6.8.0

unit.transfer.order

- **Modul**: cmas-dwh-server
- **Beschreibung**: Legt fest, in welcher Reihenfolge Datenobjektgruppen an das DWH übertragen werden.
- **Typ**: string
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: ja
- **Beispielwert**: company;customer
- **Seit**: 6.6.19
- **Entfernt in**: 6.8.1

G.2.3.2 Konfiguration von Indexer und Suche

Indexer

big.task.minimum.size

- **Modul**: cmas-core-index-common
- **Beschreibung**: Gibt die Minimalgröße eines Index-Tasks an (in Teilen, jeder Teil hat 100 Einheiten), um diesen Task als einen großen Task zu qualifizieren. Große Tasks haben eine niedrigere Priorität als normale.
- **Typ**: integer
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: 15 (Standardwert)
- **Seit**: 6.8.3

database.notification.enabled

- **Modul**: cmas-core-index-common
- **Beschreibung**: Legt fest, ob statt JMS der Database Notification Channel der Index-Aktualisierung verwendet werden soll.
- **Typ**: boolean

- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: false
- **Seit**: 6.8.4.7

database.notification.redelivery.delay.seconds

- **Modul**: cmas-core-index-common
- **Beschreibung**: Legt bei Verwendung des Database Notification Channel der Index-Aktualisierung fest, mit welcher Verzögerung die Benachrichtigung erneut gesendet wird, wenn eine Exception auftritt.
- **Typ**: integer
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: 60
- **Seit**: 6.8.4.7

database.notification.redelivery.max.attempts

- **Modul**: cmas-core-index-common
- **Beschreibung**: Legt bei Verwendung des Database Notification Channel der Index-Aktualisierung fest, wie oft maximal versucht wird, die Benachrichtigung erneut zu senden, wenn eine Exception auftritt.
- **Typ**: integer
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: 60
- **Seit**: 6.8.4.7

disable.admin.task.auto.commit

- **Modul**: cmas-core-index-common
- **Beschreibung**: Alle Tasks, die für eine Index-Aktualisierung erstellt werden, werden automatisch direkt nach ihrer Erstellung ausgeführt.
- **Typ**: boolean
- **Neustart erforderlich**: nein
- **System**: ja

- **Optional**: nein
- **Beispielwert**: false
- **Seit**: 6.6.1

index.attachment

- **Modul**: cmas-core-index-common
- **Beschreibung**: Beschreibt, ob der Inhalt von Attachments indiziert wird.
- **Typ**: boolean
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: true
- **Seit**: 6.4.3

index.history

- **Modul**: cmas-core-index-common
- **Beschreibung**: Legt fest, ob das Unit- und das Ticketprotokoll indiziert werden.
- **Typ**: boolean
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: false
- **Seit**: 6.1.0
- **Entfernt in**: 6.11.0

index.status

- **Modul**: cmas-core-index-common
- **Beschreibung**: Status des Indexers, mögliche Werte sind RED, YELLOW, GREEN, werden im Admin Tool angezeigt.
- **Typ**: string
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: GREEN
- **Seit**: 6.6.1

index.task.worker.threads

- **Modul**: cmas-core-index-common
- **Beschreibung**: Legt fest, wie viele Threads benutzt werden, um Index-Aufgaben auszuführen (Synchronisierung, Administrations- und Reparatur-Aufgaben).
- **Typ**: integer
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: 1 (Standardwert) (wir empfehlen einen Wert zu verwenden, der nicht größer als 2 ist)
- **Seit**: 6.6.14, 6.7.3. Seit 6.8.0 und auschließlich in 6.6.21 sind auch normale (live) Index-Aktualisierungen von dieser Property betroffen.

index.version.current

- **Modul**: cmas-core-index-common
- **Beschreibung**: Enthält Informationen über die derzeitige (möglicherweise veraltete) Index-version.
- **Typ**: integer
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: 1 (Standardwert)
- **Seit**: 6.7.0

index.version.newest

- **Modul**: cmas-core-index-common
- **Beschreibung**: Enthält Informationen, welche Indexversion als die neueste betrachtet wird.
- **Typ**: integer
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: 1 (Standardwert)
- **Seit**: 6.7.0

indexed.assets.per.thread.in.memory

- **Modul**: cmas-core-index-common
- **Beschreibung**: Legt fest, wie viele Assets während des Indizierens pro Thread auf einmal in den Speicher geladen werden.
- **Typ**: integer
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: 200 (Standardwert)
- **Seit**: 6.8.0

indexed.engineers.per.thread.in.memory

- **Modul**: cmas-core-index-common
- **Beschreibung**: Legt fest, wie viele Bearbeiter während des Indizierens pro Thread auf einmal in den Speicher geladen werden.
- **Typ**: integer
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: 300 (Standardwert)
- **Seit**: 6.6.14, 6.7.3

indexed.resources.per.thread.in.memory

- **Modul**: cmas-core-index-common
- **Beschreibung**: Legt fest, wie viele Ressourcen während des Indizierens pro Thread auf einmal in den Speicher geladen werden.
- **Typ**: integer
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: 200 (Standardwert)
- **Seit**: 6.10.0.0

indexed.tickets.per.thread.in.memory

- **Modul**: cmas-core-index-common
- **Beschreibung**: Legt fest, wie viele Tickets während des Indizierens pro Thread auf einmal in den Speicher geladen werden.

- **Typ**: integer
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: 100 (Standardwert)
- **Seit**: 6.6.14, 6.7.3

indexed.units.per.thread.in.memory

- **Modul**: cmas-core-index-common
- **Beschreibung**: Legt fest, wie viele Units während des Indizierens pro Thread auf einmal in den Speicher geladen werden.
- **Typ**: integer
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: 200 (Standardwert)
- **Seit**: 6.6.14, 6.7.3

synchronize.master.address

- **Modul**: cmas-core-index-common
- **Beschreibung**: Wert der Java-System-Property *-Dcmas.http.host.port*, die angibt, unter welcher URL der Index-Master-Server erreichbar ist. Standard ist Null. Seit CM-Version 6.6.17 ist dieser Wert beim Setup konfigurierbar, um den initialen Index-Master-Server zu bestimmen. Bitte beachten Sie, dass das Verändern dieses Wertes nur erlaubt ist, wenn alle Cluster-Nodes zum Empfang von Index-Veränderungen gestoppt sind.
- **Typ**: integer
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: ja
- **Beispielwert**: 127.0.0.1:80
- **Seit**: 6.6.0

synchronize.master.security.token

- **Modul**: cmas-core-index-common
- **Beschreibung**: Das Passwort für den URL-Zugriff auf den Index-Snapshot, z. B. für die Index-Synchronisierung oder für Backups.
- **Typ**: string
- **Neustart erforderlich**: nein

- **System**: ja
- **Optional**: ja
- **Beispielwert**: token
- **Seit**: 6.6.0

synchronize.master.security.user

- **Modul**: cmas-core-index-common
- **Beschreibung**: Der Benutzername für den URL-Zugriff auf den Index-Snapshot, z. B. für die Index-Synchronisierung oder für Backups.
- **Typ**: string
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: ja
- **Beispielwert**: user
- **Seit**: 6.6.0

synchronize.master.timeout.minutes

- **Modul**: cmas-core-index-common
- **Beschreibung**: Legt fest, wie oft die Index-Synchronisierung ausgehend vom aktuellen Master-Server fehlschlagen darf, bis ein neuer Master für die Index-Reparatur ausgewählt wird. Standardwert ist 5. Seit CM-Version 6.6.17 ist dieser Wert im Setup konfigurierbar, wobei 0 bedeutet, dass der Master-Server nie geändert wird (Failover-Mechanismus deaktiviert).
- **Typ**: integer
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: 5
- **Seit**: 6.6.0

synchronize.megabits.per.second

- **Modul**: cmas-core-index-common
- **Beschreibung**: Legt fest, wie viel Bandbreite der Master-Server verbrauchen darf, um Index-Änderungen an die Slave-Server zu übermitteln. Standardwert ist 85. Nutzen Sie nicht die gesamte verfügbare Bandbreite, um die Index-Änderungen zwischen den Hosts zu übermitteln, da dies dafür sorgen kann, dass die Nodes des Clusters nicht mehr synchron sind.
- **Typ**: integer
- **Neustart erforderlich**: nein
- **System**: ja

- **Optional**: nein
- **Beispielwert**: 85
- **Seit**: 6.6.0

synchronize.sleep.millis

- **Modul**: cmas-core-index-common
- **Beschreibung**: Legt fest, wie oft jeder Slave-Server den Master-Server auf Änderungen am Index abfragt. Standardwert ist 1000.
- **Typ**: integer
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: 1000
- **Seit**: 6.6.0

Search Results

globalSearchResultSizeLimit

- **Modul**: cmweb-server-adapter
- **Beschreibung**: Maximale Anzahl an Ergebnissen in der Schnellsuche.
- **Typ**: integer
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: 10
- **Seit**: 6.0

searchPageSize

- **Modul**: cmweb-server-adapter
- **Beschreibung**: Standardgröße der Seiten für Suchergebnisse.
- **Typ**: integer
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: 20
- **Seit**: 6.0

searchPageSizeOptions

- **Modul**: cmweb-server-adapter
- **Beschreibung**: Optionen für die Größe der Seiten für Suchergebnisse.
- **Typ**: string
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: 10|20|30|40|50|75|100
- **Seit**: 6.0

unitIndexSearchResultSizeLimit

- **Modul**: cmweb-server-adapter
- **Beschreibung**: Maximale Anzahl der Units in der Ergebnisliste, wenn nach Units gesucht wird (Beispiel: Kontaktsuche).
- **Typ**: integer
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: 5
- **Seit**: 6.0

G.2.3.3 LDAP-Konfiguration

LDAP-Konfiguration (wenn LDAP als Authentifizierungsmethode im CM Web Client verwendet wird)

LDAP-Parameter sind nur dann wirksam, wenn die Authentifizierungsmethode für den CM Web Client auf *LDAP* gesetzt wurde:

authentication.method

- **Modul**: cmas-core-security
- **Description**: Methode der Bearbeiter-Authentifizierung für den Web Client (interne CM-Datenbank oder LDAP-Authentifizierung). Mögliche Werte sind LDAP oder DATABASE.
- **Typ**: string
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: DATABASE
- **Seit**: 6.0

ldap.authentication

- **Modul**: cmas-core-security
- **Description**: Verwendete Authentifizierungsmethode, wenn LDAP-Authentifizierung benutzt wird.
- **Typ**: string
- **Neustart erforderlich**: ja
- **System**: ja
- **Optional**: nein
- **Beispielwert**: simple
- **Seit**: 6.0

ldap.basedn

- **Modul**: cmas-core-security
- **Beschreibung**: Base DN für die Suche von LDAP-Benutzerkonten, wenn LDAP-Authentifizierung verwendet wird.
- **Typ**: string
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: ou=accounts,dc=consol,dc=de
- **Seit**: 6.0

ldap.initialcontextfactory

- **Modul**: cmas-core-security
- **Beschreibung**: Name der Java-Klasse für die Initial Context Factory der LDAP-Implementierung bei der Verwendung von LDAP-Authentifizierung. Ist üblicherweise *com.sun.jndi.ldap.LdapCtxFactory*.
- **Typ**: string
- **Neustart erforderlich**: ja
- **System**: ja
- **Optional**: nein
- **Beispielwert**: com.sun.jndi.ldap.LdapCtxFactory
- **Seit**: 6.0

ldap.password

- **Modul**: cmas-core-security
- **Beschreibung**: Passwort für die Verbindung zum LDAP, um Benutzer zu suchen, wenn LDAP-Authentifizierung verwendet wird. Wird nur benötigt, wenn die Suche nicht anonym ausgeführt werden kann.
- **Typ**: password
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: ja
- **Seit**: 6.1.2

ldap.providerurl

- **Modul**: cmas-core-security
- **Beschreibung**: LDAP-Provider, wenn LDAP-Authentifizierung verwendet wird.
- **Typ**: string
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: ldap://myserver.consol.de:389
- **Seit**: 6.0

ldap.searchattr

- **Modul**: cmas-core-security
- **Beschreibung**: Suchattribut für die Suche nach LDAP-Einträgen, die mit dem CM-Login verbunden sind.
- **Typ**: string
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: uid
- **Seit**: 6.0

ldap.userdn

- **Modul**: cmas-core-security
- **Beschreibung**: LDAP-Benutzer für die Verbindung zum LDAP, um Benutzer zu suchen, wenn LDAP-Authentifizierung verwendet wird. Wird nur benötigt, wenn die Suche nicht anonym ausgeführt werden kann.

- **Typ**: string
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: ja
- **Seit**: 6.1.2

LDAP-Konfiguration (wenn LDAP in CM.Track als Authentifizierungsmethode eingesetzt wird)

LDAP-Parameter sind nur dann wirksam, wenn die Authentifizierungsmethode für CM.Track auf *LDAP* gesetzt wurde:

contact.authentication.method

- **Modul**: cmas-core-security
- **Beschreibung**: Definiert die Kontakt-Authentifizierungsmethode für CM.Track, mögliche Werte sind DATABASE oder LDAP oder LDAP,DATABASE oder DATABASE,LDAP.
- **Typ**: string
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Seit**: 6.9.3.0

ldap.contact.name.basedn

- **Modul**: cmas-core-security
- **Description**: Base DN für die Suche nach der Kontakt-DN mittels LDAP-ID (z. B. ou=a-ccounts,dc=consol,dc=de).
- **Typ**: string
- **Neustart erforderlich**: nein
- **System**: nein
- **Optional**: ja
- **Seit**: 6.9.3.0

ldap.contact.name.password

- **Modul**: cmas-core-security
- **Beschreibung**: Passwort für die Suche nach der Kontakt-DN mittels LDAP-ID. Wenn nicht gesetzt, wird ein anonymes Konto verwendet.
- **Typ**: string
- **Neustart erforderlich**: nein
- **System**: nein

- **Optional**: ja
- **Seit**: 6.9.3.0

ldap.contact.name.providerurl

- **Modul**: cmas-core-security
- **Beschreibung**: Adresse des LDAP-Servers (ldap[s]://host:port).
- **Typ**: string
- **Neustart erforderlich**: nein
- **System**: nein
- **Optional**: ja
- **Seit**: 6.9.3.0

ldap.contact.name.searchattr

- **Modul**: cmas-core-security
- **Beschreibung**: Attribut für die Suche nach der Kontakt-DN mittels LDAP-ID (z. B. uid).
- **Typ**: string
- **Neustart erforderlich**: nein
- **System**: nein
- **Optional**: ja
- **Seit**: 6.9.3.0

ldap.contact.name.userdn

- **Modul**: cmas-core-security
- **Beschreibung**: Benutzer-DN für die Suche nach Kontakt-DN mittels LDAP-ID. Wenn nicht gesetzt, wird ein anonymes Konto verwendet.
- **Typ**: string
- **Neustart erforderlich**: nein
- **System**: nein
- **Optional**: ja
- **Seit**: 6.9.3.0

ldap.initialcontextfactory

- **Modul**: cmas-core-security
- **Beschreibung**: Name der Java-Klasse für die Initial Context Factory der LDAP-Implementierung bei der Verwendung von LDAP-Authentifizierung. Ist üblicherweise *com.sun.jndi.ldap.LdapCtxFactory*.
- **Typ**: string
- **Neustart erforderlich**: ja

- **System**: ja
- **Optional**: nein
- **Beispielwert**: com.sun.jndi.ldap.LdapCtxFactory
- **Seit**: 6.0

G.2.3.4 E-Mail-Konfiguration

Postausgang

Unabhängig vom Modus für eingehende E-Mails (Mule/ESB und NIMH).

mail.smtp.email

- **Modul**: cmas-core-server
- **Beschreibung**: SMTP-E-Mail-URL für ausgehende E-Mails.
- **Typ**: string
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: smtp://mail.mydomain.com:25
- **Seit**: 6.0

mail.smtp.envelopesender

- **Modul**: cmas-core-server
- **Beschreibung**: E-Mail-Adresse, die als Absender im SMTP-Envelope benutzt wird. Wenn nicht gesetzt, wird die From-Adresse der E-Mail benutzt.
- **Typ**: string
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: mysender@mydomain.com
- **Seit**: 6.5.7

mail.from

- **Modul**: cmweb-server-adapter
- **Beschreibung**: Diese E-Mail-Adresse wird anstelle der E-Mail-Adresse des Bearbeiters in E-Mail-Konversationen verwendet.
- **Typ**: string
- **Neustart erforderlich**: nein
- **System**: ja

- **Optional**: ja
- **Seit**: 6.1.2

mail.reply.to

- **Modul**: cmweb-server-adapter
- **Beschreibung**: Wenn dieser Wert gesetzt ist, zeigt der Web Client diesen Wert beim Versenden einer E-Mail im Reply-To-Feld an.
- **Typ**: string
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: ja
- **Seit**: 6.0.1

> ⚠ Bitte lesen Sie dazu die ausführlichen Informationen über Reply-To-Adressen in ConSol CM im Abschnitt *Skripte des Typs E-Mail* im *ConSol CM Administratorhandbuch*.

mailTemplateAboveQuotedText

- **Modul**: cmweb-server-adapter
- **Beschreibung**: Legt das Verhalten der E-Mail-Vorlagen im Ticket-E-Mail-Editor fest, wenn eine andere E-Mail zitiert wird, d. h. auf diese geantwortet oder diese weitergeleitet wird. Wird oft verwendet, um die Signatur korrekt zu platzieren.
- **Typ**: boolean
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: false
- **Seit**: 6.2.4

mail.sender.address

- **Modul**: cmas-workflow-jbpm
- **Beschreibung**: From-Adresse für E-Mails, die von der Workflow-Engine versendet werden.
- **Typ**: string
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: myuser@consol.de

- **Entfernt in**: 6.8.0
- **Ersetzt durch**: jobExecutor.mailFrom

Posteingang

Einstellungen für Mule/ESB

esb.directory

- **Modul**: cmas-esb-core
- **Beschreibung**: Von Mule/ESB verwendetes Verzeichnis.
- **Typ**: string
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: C:\Users\user\cmas\mule
- **Seit**: 6.0
- **Entfernt in**: 6.11.0

mail.attachments.validation.info.sender

- **Modul**: cmas-esb-mail
- **Beschreibung**: Setzt den From-Header bei Attachments vom Typ *error notification mail*. Standardmäßig wird die E-Mail-Adresse des Administrators verwendet, die bei der Systeminstallation angegeben wurde.
- **Typ**: string
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: admin@consolcm.com
- **Seit**: 6.7.5
- **Entfernt in**: 6.11.0

mail.attachments.validation.info.subject

- **Modul**: cmas-esb-mail
- **Beschreibung**: Setzt den Betreff bei Attachments vom Typ *error notification mail*.
- **Typ**: string
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein

- **Beispielwert**: E-Mail konnte nicht verarbeitet werden, weil ihre Attachments zurückgewiesen wurden!
- **Seit**: 6.7.5
- **Entfernt in**: 6.11.0

mail.callname.pattern

- **Modul**: cmas-esb-mail
- **Beschreibung**: Regulärer Ausdruck für den Betreff von eingehenden E-Mails. Verfügbar als TICKET_NAME_PATTERN_FORMAT in Skripten für eingehende E-Mails.
- **Typ**: string
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: .*?Ticket\s+\((\S+)\).*
- **Seit**: 6.0
- **Entfernt in**: 6.11.0

mail.cluster.node.id

- **Modul**: cmas-esb-mail
- **Beschreibung**: Nur der Node, dessen *mail.cluster.node.id* gleich *cmas.clusternode.id* ist, startet den Mule/ESB E-Mail-Service.
- **Typ**: string
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: unspecified
- **Seit**: 6.6.5
- **Entfernt in**: 6.11.0

mail.db.archive

- **Modul**: cmas-esb-mail
- **Beschreibung**: Wenn dieser Wert auf *true* gesetzt ist, werden eingehende E-Mails in der Datenbank archiviert.
- **Typ**: boolean
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: ja

- **Beispielwert**: false (Standardwert)
- **Seit**: 6.8.5.5
- **Entfernt in**: 6.11.0

Obsolet! Im Mule/ESB-Modus werden keine E-Mails in der Datenbank gespeichert. E-Mails, die nicht verarbeitet werden können, werden im Dateisystem gespeichert, siehe Abschnitt *E-Mail-Backups* im *ConSol CM Administratorhandbuch*.

mail.delete.read

- **Modul**: cmas-esb-mail
- **Beschreibung**: Legt fest, ob CM die per IMAP(S) abgeholten E-Mails löscht. Wenn der Wert auf *true* gesetzt wird, werden die E-Mails nach der Abholung gelöscht. Standardmäßig werden die per IMAP(S) abgeholten E-Mails nicht gelöscht. Hinweis: E-Mails, die per POP3(S) abgeholt werden, werden immer gelöscht.
- **Typ**: boolean
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: true
- **Seit**: 6.7.3
- **Entfernt in**: 6.11.0

mail.incoming.uri

- **Modul**: cmas-esb-mail
- **Beschreibung**: URL für eingehende E-Mails.
- **Typ**: string
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: pop3://cm-incoming-user:password@localhost:10110
- **Seit**: 6.0
- **Entfernt in**: 6.11.0

Dieser Wert sollte nicht innerhalb des System-Property-Fensters verändert werden. Die Postfächer sollten stattdessen im Admin Tool im Navigationselement *E-Mail* konfiguriert werden. Wenn Sie dieses Element für die Konfiguration benutzen, können Sie alle Einträge kontrolliert konfigurieren, d. h. für jedes Postfach, das hinzugefügt wird, baut CM während der Einrichtung eine Testverbindung auf. Auf diese Weise ist es nicht möglich, falsche Werte einzugeben.

mail.max.restarts

- **Modul**: cmas-esb-mail
- **Beschreibung**: Maximale Anzahl der Neustarts des E-Mail-Services, bevor aufgegeben wird.
- **Typ**: integer
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: 3
- **Seit**: 6.0
- **Entfernt in**: 6.11.0

mail.mime.strict

- **Modul**: cmas-esb-mail
- **Beschreibung**: Wenn dieser Wert auf *false* gesetzt ist, werden E-Mail-Adressen nicht auf strikte MIME-Übereinstimmung geparst. Standard ist *true*, was bedeutet, dass auf strikte MIME-Übereinstimmung geprüft wird.
- **Typ**: boolean
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: false
- **Seit**: 6.6.17, 6.7.3
- **Entfernt in**: 6.11.0

mail.mule.service

- **Modul**: cmas-esb-mail
- **Beschreibung**: From-Adresse für E-Mails, die vom Mule-Service aus gesendet werden.
- **Typ**: email
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: myuser@consol.de
- **Seit**: 6.0
- **Entfernt in**: 6.11.0

mail.polling.interval

- **Modul**: cmas-esb-mail
- **Beschreibung**: Abrufintervall für E-Mails in Millisekunden.
- **Typ**: integer
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: 60000
- **Seit**: 6.0
- **Entfernt in**: 6.11.0

mail.process.error

- **Modul**: cmas-esb-mail
- **Beschreibung**: To-Adresse für E-Mails mit Fehlermeldungen des Mule. Standardmäßig wird die E-Mail-Adresse des Administrators verwendet, die bei der Systeminstallation angegeben wurde.
- **Typ**: email
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: myuser@consol.de
- **Seit**: 6.0
- **Entfernt in**: 6.11.0

mail.process.retry.attempts

- **Modul**: cmas-esb-mail
- **Beschreibung**: Anzahl der erneuten Versuche, wenn E-Mails verarbeitet werden.
- **Typ**: integer
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: 3
- **Seit**: 6.0.2
- **Entfernt in**: 6.11.0

mail.process.timeout

- **Modul**: cmas-esb-mail
- **Beschreibung**: Timeout für die E-Mail-Verarbeitung in Sekunden.
- **Typ**: integer
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: 60
- **Seit**: 6.1.3
- **Entfernt in**: 6.11.0

mail.redelivery.retry.count

- **Modul**: cmas-esb-mail
- **Beschreibung**: Gibt die Anzahl der Versuche an, eine E-Mail aus dem CM-System erneut zuzustellen.
- **Typ**: integer
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: 3
- **Seit**: 6.1.0
- **Entfernt in**: 6.11.0

Einstellungen für NIMH

Diese Einstellungen sind wirksam, wenn NIMH aktiviert (und daher Mule/ESB deaktiviert) ist:

nimh.enabled

- **Modul**: cmas-core-server
- **Beschreibung**: Aktiviert den NIMH-Dienst. Im Cluster muss die Node-ID angehängt werden, z. B. *nimh.enabled.NODEID = true*.
- **Typ**: boolean
- **Neustart erforderlich**: nein
- **System**: nein
- **Optional**: ja
- **Beispielwert**: false
- **Seit**: 6.9.4.0

filesystem.polling.threads.number

- **Modul**: cmas-nimh
- **Beschreibung**: Anzahl der Threads, die für die Abfrage der Datenbank-E-Mail-Warteschlange gestartet werden. Standardwert: 1.
- **Typ**: integer
- **Neustart erforderlich**: nein
- **System**: nein
- **Optional**: ja
- **Beispielwert**: 10
- **Seit**: 6.4.0

filesystem.polling.threads.shutdown.timeout.seconds

- **Modul**: cmas-nimh
- **Beschreibung**: Wartezeit nach dem Shutdown-Signal. Wenn der Timeout erreicht ist, wird der Thread beendet. Standardwert: 60.
- **Typ**: integer
- **Neustart erforderlich**: nein
- **System**: nein
- **Optional**: ja
- **Beispielwert**: 60
- **Seit**: 6.4.0

filesystem.polling.threads.watchdog.interval.seconds

- **Modul**: cmas-nimh
- **Beschreibung**: Intervall des Watchdog Threads. Standardwert: 30.
- **Typ**: integer
- **Neustart erforderlich**: nein
- **System**: nein
- **Optional**: ja
- **Beispielwert**: 60
- **Seit**: 6.4.0

filesystem.task.enabled

- **Modul**: cmas-nimh
- **Beschreibung**: Mit dieser System-Property kann der Service Thread eines bestimmten Pollers deaktiviert werden. Standardwert: true.
- **Typ**: boolean

- **Neustart erforderlich**: nein
- **System**: nein
- **Optional**: ja
- **Beispielwert**: true
- **Seit**: 6.4.0

filesystem.task.interval.seconds

- **Modul**: cmas-nimh
- **Beschreibung**: Standard-Intervall für den Abruf von Postfächern. Standardwert: 60 Sekunden
- **Typ**: integer
- **Neustart erforderlich**: nein
- **System**: nein
- **Optional**: ja
- **Beispielwert**: 60
- **Seit**: 6.4.0

filesystem.task.polling.folder

- **Modul**: cmas-nimh
- **Beschreibung**: Speicherort des Ordners, der überwacht und nach E-Mails im Format von eml-Dateien durchsucht wird. Standard: Unterverzeichnis "mail" des cmas-Datenverzeichnisses
- **Typ**: string
- **Neustart erforderlich**: nein
- **System**: nein
- **Optional**: ja
- **Beispielwert**: c://cmas//mail
- **Seit**: 6.4.0

filesystem.task.timeout.seconds

- **Modul**: cmas-nimh
- **Beschreibung**: Nach dieser Zeit (der Inaktivität) wird der Service Thread als beschädigt betrachtet und automatisch neu gestartet. Standardwert: 120 Sekunden
- **Typ**: integer
- **Neustart erforderlich**: nein
- **System**: nein
- **Optional**: ja
- **Beispielwert**: 60
- **Seit**: 6.4.0

filesystem.task.transaction.timeout.seconds

- **Modul**: cmas-nimh
- **Beschreibung**: Standard-Transaktions-Timeout für Transaktionen, die E-Mails abrufen. Sollte mit der Anzahl der Nachrichten, die gleichzeitig abgeholt werden, korrelieren. Standardwert: 60 Sekunden
- **Typ**: integer
- **Neustart erforderlich**: nein
- **System**: nein
- **Optional**: ja
- **Beispielwert**: 60
- **Seit**: 6.4.0

mailbox.1.connection.host

- **Modul**: cmas-nimh
- **Beschreibung**: Host (Server) für das erste konfigurierte Postfach. Überschreibt den Standardparameter *mailbox.default.connection.host*.

mailbox.1.connection.password

- **Modul**: cmas-nimh
- **Beschreibung**: Passwort für das erste konfigurierte Postfach. Überschreibt den Standardparameter *mailbox.default.connection.password*.

mailbox.1.connection.port

- **Modul**: cmas-nimh
- **Beschreibung**: Port für das erste konfigurierte Postfach. Überschreibt den Standardparameter *mailbox.default.connection.port*.

mailbox.1.connection.protocol

- **Modul**: cmas-nimh
- **Beschreibung**: Protokoll (z. B. IMAP oder POP3) für das erste konfigurierte Postfach. Überschreibt den Standardparameter *mailbox.default.connection.protocol*.

mailbox.1.connection.username

- **Modul**: cmas-nimh
- **Beschreibung**: Benutzername für das erste konfigurierte Postfach. Überschreibt den Standardparameter *mailbox.default.connection.username*.

mailbox.2.connection.host

- **Modul**: cmas-nimh
- **Beschreibung**: Host (Server) für das zweite konfigurierte Postfach. Überschreibt den Standardparameter *mailbox.default.connection.host*.

mailbox.2.connection.password

- **Modul**: cmas-nimh
- **Beschreibung**: Passwort für das zweite konfigurierte Postfach. Überschreibt den Standardparameter *mailbox.default.connection.password*.

mailbox.2.connection.port

- **Modul**: cmas-nimh
- **Beschreibung**: Port für das zweite konfigurierte Postfach. Überschreibt den Standardparameter *mailbox.default.connection.port*.

mailbox.2.connection.protocol

- **Modul**: cmas-nimh
- **Beschreibung**: Protokoll (z. B. IMAP oder POP3) für das zweite konfigurierte Postfach. Überschreibt den Standardparameter *mailbox.default.connection.protocol*.

mailbox.2.connection.username

- **Modul**: cmas-nimh
- **Beschreibung**: Benutzername für das zweite konfigurierte Postfach. Überschreibt den Standardparameter *mailbox.default.connection.username*.

Für alle Postfach-Properties im Zusammenhang mit NIMH gilt folgendes Prinzip: Es wird eine Standard-Property definiert (z.B *mailbox.default.connection.port*). Wenn keine postfachspezifische Property konfiguriert ist, wird dieser Standardwert verwendet.

mailbox.default.connection.host

- **Modul**: cmas-nimh
- **Beschreibung**: Host (Servername) eines bestimmten Postfachs, aus dem der Poller E-Mails liest.
- **Typ**: string
- **Neustart erforderlich**: nein
- **System**: nein
- **Optional**: ja
- **Beispielwert**: 10.10.1.157
- **Seit**: 6.4.0

mailbox.default.connection.password

- **Modul**: cmas-nimh
- **Beschreibung**: Passwort für ein bestimmtes Postfach, aus dem der Poller E-Mails liest.
- **Typ**: string
- **Neustart erforderlich**: nein
- **System**: nein
- **Optional**: ja
- **Beispielwert**: consol
- **Seit**: 6.4.0

mailbox.default.connection.port

- **Modul**: cmas-nimh
- **Beschreibung**: Port für ein bestimmtes Postfach, aus dem der Poller E-Mails liest.
- **Typ**: string
- **Neustart erforderlich**: nein
- **System**: nein
- **Optional**: ja
- **Beispielwert**: 143
- **Seit**: 6.4.0

mailbox.default.connection.protocol

- **Modul**: cmas-nimh
- **Beschreibung**: Protokoll des Pollers, z. B. IMAP oder POP3. Kein Standardwert.
- **Typ**: string
- **Neustart erforderlich**: nein
- **System**: nein
- **Optional**: ja
- **Beispielwert**: imap
- **Seit**: 6.4.0

mailbox.default.connection.username

- **Modul**: cmas-nimh
- **Beschreibung**: Benutzername für ein bestimmtes Postfach, aus dem der Poller E-Mails liest.
- **Typ**: string
- **Neustart erforderlich**: nein
- **System**: nein

- **Optional**: ja
- **Beispielwert**: username
- **Seit**: 6.4.0

mailbox.default.session.mail.debug

- **Modul**: cmas-nimh
- **Beschreibung**: Beispiel für javax.mail-Property - ermöglicht detaillierteres Session-Debugging mit javax.mail.
- **Typ**: boolean
- **Neustart erforderlich**: nein
- **System**: nein
- **Optional**: ja
- **Beispielwert**: true
- **Seit**: 6.4.0

mailbox.default.session.mail.imap.timeout

- **Modul**: cmas-nimh
- **Beschreibung**: Beispiel für javax.mail-Property
- **Typ**: integer
- **Neustart erforderlich**: nein
- **System**: nein
- **Optional**: ja
- **Beispielwert**: 120
- **Seit**: 6.4.0

mailbox.default.session.mail.mime.address.strict

- **Modul**: cmas-nimh
- **Beschreibung**: Beispiel für javax.mail-Property - Gegenstück zum alten *mail.mime.strict* von Mule, erlaubt das nicht so strikte Parsen des E-Mail-Headers.
- **Typ**: boolean
- **Neustart erforderlich**: nein
- **System**: nein
- **Optional**: ja
- **Beispielwert**: true
- **Seit**: 6.4.0

mailbox.default.session.mail.pop3.timeout

- **Modul**: cmas-nimh
- **Beschreibung**: Beispiel für javax.mail-Property
- **Typ**:
- **Neustart erforderlich**:
- **System**:
- **Optional**:
- **Beispielwert**:
- **Seit**: 6.4.0

mailbox.default.session.mail.<protocol>.partialfetch

- **Modul**: cmas-nimh
- **Beschreibung**: z. B. mailbox.default.session.mail.imaps.partialfetch
- **Typ**: boolean
- **Neustart erforderlich**: nein
- **System**: nein
- **Optional**: ja
- **Beispielwert**: false
- **Seit**:

mailbox.default.task.delete.read.messages

- **Modul**: cmas-nimh
- **Beschreibung**: Legt fest, ob Nachrichten nach dem Lesen aus dem Postfach entfernt werden sollen. Beim IMAP-Protokoll werden Nachrichten standardmäßig als SEEN gekennzeichnet. Bei POP3 wird die Nachricht nur gelöscht, wenn der Wert auf true gesetzt ist. Anderenfalls bleibt die Nachricht auf dem Server, was zu einer Endlos-Leseschleife führt. Standardwert: false.
- **Typ**: boolean
- **Neustart erforderlich**: nein
- **System**: nein
- **Optional**: ja
- **Beispielwert**: false
- **Seit**: 6.4.0

mailbox.default.task.enabled

- **Modul**: cmas-nimh
- **Beschreibung**: Mit dieser System-Property kann der Service Thread eines bestimmten Pollers deaktiviert werden. Standardwert: true.

- **Typ**: boolean
- **Neustart erforderlich**: nein
- **System**: nein
- **Optional**: ja
- **Beispielwert**: false
- **Seit**: 6.4.0

mailbox.default.task.interval.seconds

- **Modul**: cmas-nimh
- **Beschreibung**: Standard-Intervall für den Abruf von Postfächern. Standardwert: 60 Sekunden
- **Typ**: integer
- **Neustart erforderlich**: nein
- **System**: nein
- **Optional**: ja
- **Beispielwert**: 60
- **Seit**: 6.4.0

mailbox.default.task.max.message.size

- **Modul**: cmas-nimh
- **Beschreibung**: Maximale Größe von E-Mails (d. h. E-Mail plus Attachments). E-Mails, die größer als dieser Wert sind, werden nicht automatisch von NIMH verarbeitet. Sie werden in der Datenbank (Tabelle *cmas_nimh_archived_mail*) gespeichert und erscheinen daher in den E-Mail-Backups im Admin Tool (siehe Abschnitt *E-Mail-Backups* im *ConSol CM Administratorhandbuch*). Von dort können sie erneut gesendet, in das Dateisystem geladen oder gelöscht werden. Für diese Operationen ist die Größe der Nachricht nicht relevant. Standardmäßig auf 10 MB (10485760) gesetzt.
- **Typ**: integer
- **Neustart erforderlich**: nein
- **System**: nein
- **Optional**: ja
- **Beispielwert**: 10485760
- **Seit**: 6.4.0

mailbox.default.task.max.messages.per.run

- **Modul**: cmas-nimh
- **Beschreibung**: Anzahl der Nachrichten, die gleichzeitig aus dem Postfach abgeholt werden. Muss mit dem Transaktions-Timeout korrelieren. Standardwert: 20.
- **Typ**: integer

- **Neustart erforderlich**: nein
- **System**: nein
- **Optional**: ja
- **Beispielwert**: 60
- **Seit**: 6.4.0

mailbox.default.task.timeout.seconds

- **Modul**: cmas-nimh
- **Beschreibung**: Nach dieser Zeit (der Inaktivität) wird der Service Thread als beschädigt betrachtet und automatisch neu gestartet. Standardwert: 120 Sekunden
- **Typ**: integer
- **Neustart erforderlich**: nein
- **System**: nein
- **Optional**: ja
- **Beispielwert**: 60
- **Seit**: 6.4.0

mailbox.default.task.transaction.timeout.seconds

- **Modul**: cmas-nimh
- **Beschreibung**: Standard-Transaktions-Timeout für Transaktionen, die E-Mails abrufen. Sollte mit der Anzahl der Nachrichten, die gleichzeitig abgeholt werden, korrelieren. Standardwert: 60 Sekunden
- **Typ**: integer
- **Neustart erforderlich**: nein
- **System**: nein
- **Optional**: ja
- **Beispielwert**: 60
- **Seit**: 6.4.0

mailbox.polling.threads.mail.log.enabled

- **Modul**: cmas-nimh
- **Description**: Ermöglicht das Protokollieren von E-Mails, was in Cluster-Umgebungen besonders wichtig ist (wird dort als Semaphor genutzt).
- **Typ**: boolean
- **Neustart erforderlich**: nein
- **System**: nein
- **Optional**: ja

- **Beispielwert**: true (Standardwert)
- **Seit**: 6.9.4.1

mailbox.polling.threads.number

- **Modul**: cmas-nimh
- **Beschreibung**: Anzahl der Threads für den Zugriff auf Postfächer. Standardwert: 1.
- **Typ**: integer
- **Neustart erforderlich**: nein
- **System**: nein
- **Optional**: ja
- **Beispielwert**: 1
- **Seit**: 6.4.0

queue.polling.threads.number

- **Modul**: cmas-nimh
- **Beschreibung**: Anzahl der Threads, die zur Überwachung der E-Mail-Warteschlange gestartet werden. Standardwert: 1.
- **Typ**: integer
- **Neustart erforderlich**: nein
- **System**: nein
- **Optional**: ja
- **Beispielwert**: 1
- **Seit**: 6.4.0

queue.polling.threads.shutdown.timeout.seconds

- **Modul**: cmas-nimh
- **Beschreibung**: Wartezeit nach dem Shutdown-Signal. Wenn der Timeout erreicht ist, wird der Thread beendet. Standardwert: 60.
- **Typ**: integer
- **Neustart erforderlich**: nein
- **System**: nein
- **Optional**: ja
- **Beispielwert**: 60
- **Seit**: 6.4.0

queue.polling.threads.watchdog.interval.seconds

- **Modul**: cmas-nimh
- **Beschreibung**: Intervall des Watchdog Threads. Standardwert: 30.
- **Typ**: integer
- **Neustart erforderlich**: nein
- **System**: nein
- **Optional**: ja
- **Beispielwert**: 30
- **Seit**: 6.4.0

queue.task.error.pause.seconds

- **Modul**: cmas-nimh
- **Beschreibung**: Anzahl der Sekunden, die der Queue-Poller nach einem Infrastrukturfehler (z. B. der Datenbank) wartet. Standardwert: 180 Sekunden
- **Typ**: integer
- **Neustart erforderlich**: nein
- **System**: nein
- **Optional**: ja
- **Beispielwert**: 180
- **Seit**: 6.4.0

queue.task.interval.seconds

- **Modul**: cmas-nimh
- **Beschreibung**: Thread-Intervall zur Überwachung der Haupt-E-Mail-Warteschlange. Standardwert: 15.
- **Typ**: integer
- **Neustart erforderlich**: nein
- **System**: nein
- **Optional**: ja
- **Beispielwert**: 15
- **Seit**: 6.4.0

queue.task.max.retries

- **Modul**: cmas-nimh
- **Beschreibung**: Maximale Anzahl der Versuche, eine E-Mail nach einer Exception erneut zu verarbeiten. Ist diese erreicht, wird die E-Mail archiviert. Die archivierte E-Mail kann über das NIMH API (oder das Admin Tool) wieder aktiviert werden.

- **Typ**: integer
- **Neustart erforderlich**: nein
- **System**: nein
- **Optional**: ja
- **Beispielwert**: 10
- **Seit**: 6.4.0

queue.task.timeout.seconds

- **Modul**: cmas-nimh
- **Beschreibung**: Nach dieser Zeit (der Inaktivität) wird der Service Thread als beschädigt betrachtet und automatisch neu gestartet. Standardwert: 600 Sekunden
- **Typ**: integer
- **Neustart erforderlich**: nein
- **System**: nein
- **Optional**: ja
- **Beispielwert**: 600
- **Seit**: 6.4.0

queue.task.transaction.timeout.seconds

- **Modul**: cmas-nimh
- **Beschreibung**: Transaktions-Timeout für die E-Mail-Verarbeitung in der Warteschlange. Standardwert: 60.
- **Typ**: integer
- **Neustart erforderlich**: nein
- **System**: nein
- **Optional**: ja
- **Beispielwert**: 60
- **Seit**: 6.4.0

mail.attachments.validation.info.sender

- **Modul**: cmas-nimh-extension
- **Beschreibung**: Setzt den From-Header bei Attachments vom Typ *error notification mail*.
- **Typ**: string
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: admin@mail.com

- **Seit**: 6.7.5

> Diese System-Property entspricht der alten *cmas-esb-mail, mail.attachments.validation.info.sender*

mail.attachments.validation.info.subject

- **Modul**: cmas-nimh-extension
- **Beschreibung**: Setzt den Betreff bei Attachments vom Typ error notification mail.
- **Typ**: string
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: E-Mail konnte nicht verarbeitet werden, weil ihre Attachments zurückgewiesen wurden!
- **Seit**: 6.7.5

> Diese System-Property entspricht der alten *cmas-esb-mail, mail.attachments.validation.info.subject*

mail.db.archive

- **Modul**: cmas-nimh-extension
- **Beschreibung**: Wenn dieser Wert auf *true* gesetzt ist, werden eingehende E-Mails in der Datenbank archiviert.
- **Typ**: boolean
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: ja
- **Beispielwert**: false (Standardwert)
- **Seit**: 6.8.5.5

mail.error.from.address

> Diese System-Property entspricht der alten *cmas-esb-mail,mail.mule.service*

mail.error.to.address

> Diese System-Property entspricht der alten *cmas-esb-mail, mail.process.error*

mail.on.error

- **Modul**: cmas-nimh-extension
- **Beschreibung**: Wenn diese Property auf *true* gesetzt ist, wird im Fall, dass eine E-Mail nicht verarbeitet werden konnte, ein Fehler-E-Mail an die oben konfigurierten E-Mail-Adressen gesendet. Standardwert: true.
- **Typ**: boolean
- **Neustart erforderlich**: nein
- **System**: nein
- **Optional**: ja
- **Beispielwert**: false
- **Seit**: 6.4.0

mail.process.error

- **Modul**: cmas-nimh-extension
- **Beschreibung**: To-Adresse für E-Mails mit Fehlermeldungen des Mule.
- **Typ**: email
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: myuser@consol.de
- **Seit**: 6.4.0

mail.ticketname.pattern

- **Modul**: cmas-nimh-extension
- **Beschreibung**: Regulärer Ausdruck, mit dem der Ticketname im Betreff eingehender E-Mails identifiziert wird.
- **Typ**: string
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: .*?Ticket\s+\((\S+)\).*
- **Seit**: 6.4.0

Zuordnung von früheren Mule- und neuen NIMH-Properties

Mule-Property	NIMH-Property
cmas-esb-mail, mail.delete.read	cmas-nimh, mailbox.default.task.delete.read.messages
cmas-esb-mail, mail.polling.interval	cmas-nimh, mailbox.default.task.interval.seconds
cmas-esb-mail, mail.process.retry.attempts	cmas-nimh, queue.task.max.retries
cmas-esb-mail, mail.mime.strict	cmas-nimh, mailbox.default.session.mail.mime.address.strict
cmas-esb-mail, mail.encryption	cmas-core-server, mail.encryption (in die core-server-Properties verschoben)
cmas-esb-mail, mail.callname.pattern	cmas-nimh-extension, mail.ticketname.pattern
cmas-esb-mail, mail.attachments.validation.info.sender	cmas-nimh-extension, mail.attachments.validation.info.sender
cmas-esb-mail, mail.attachments.validation.info.subject	cmas-nimh-extension, mail.attachments.validation.info.subject
(cmas-esb-mail, mail.db.archive)	cmas-nimh-extension, mail.db.archive
cmas-esb-mail, mail.mule.service	cmas-nimh-extension, mail.error.from.address
cmas-esb-mail, mail.process.error	cmas-nimh-extension, mail.error.to.address

Attachments für eingehende E-Mails

Diese Einstellungen gelten für Mule/ESB und NIMH.

attachment.allowed.types

- **Modul**: cmas-core-server
- **Beschreibung**: Durch Kommas getrennte Liste der erlaubten Dateinamenserweiterungen (wenn kein Werte definiert ist, sind alle Dateinamenserweiterungen erlaubt).
- **Typ**: string
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: ja

- **Beispielwert**: txt,zip,doc
- **Seit**: 6.5.0

attachment.max.size

- **Modul**: cmas-core-server
- **Beschreibung**: Maximale Größe von Attachments in MB. Dies ist eine Validierungs-Property der CM-API. Sie steuert die Größe der Attachments in Tickets, Units und Ressourcen. Außerdem steuert sie die Größe der eingehenden (nicht ausgehenden!) E-Mail-Attachments im NIMH- sowie im Mule/ESB-Modus.
- **Typ**: integer
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: 100
- **Seit**: 6.4.0

E-Mail-Verschlüsselung (Postausgang und Posteingang)

Diese Einstellungen sind nur dann wirksam, wenn die E-Mail-Verschlüsselung aktiv (true) ist. Sie gelten für Mule/ESB Mail und NIMH.

mail.encryption

- **Modul**: cmas-core-server
- **Beschreibung**: Wenn dieser Wert auf *true* gesetzt ist, ist im Ticket-E-Mail-Editor die Checkbox zur Verschlüsselung der E-Mail standardmäßig aktiviert.
- **Typ**: boolean
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: true (Standardwert = false)
- **Seit**: 6.8.4.0

Wenn Zertifikate in einem LDAP-Verzeichnis gespeichert sind, müssen folgende Einstellungen vorgenommen werden:

ldap.certificate.basedn

- **Modul**: cmas-core-server
- **Beschreibung**: Base DN für den Speicherort der Zertifikate im LDAP-Verzeichnisbaum. Wenn nicht angegeben, wird *cmas-core-security, ldap.basedn* verwendet.
- **Typ**: string
- **Neustart erforderlich**: nein

- **System**: ja
- **Optional**: ja
- **Beispielwert**: ou=accounts,dc=consol,dc=de
- **Seit**: 6.8.4

ldap.certificate.content.attribute

- **Modul**: cmas-core-server
- **Description**: Name des LDAP-Attributs, das angibt, wo Zertifikatsdaten im LDAP-Verzeichnisbaum gespeichert werden. Standardwert: usercertificate
- **Typ**: string
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: ja
- **Beispielwert**: usercertificate
- **Seit**: 6.8.4

ldap.certificate.password

- **Modul**: cmas-core-server
- **Beschreibung**: Passwort des LDAP-Zertifikatmanagers. Wenn nicht gesetzt, wird *cmas-core-security, ldap.password* verwendet.
- **Typ**: string
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: ja
- **Seit**: 6.8.4

ldap.certificate.providerurl

- **Modul**: cmas-core-server
- **Beschreibung**: URL des LDAP-Zertifikat-Providers. Wenn nicht gesetzt, wird *cmas-core-security, ldap.providerurl* verwendet.
- **Typ**: string
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: ja
- **Beispielwert**: ldap://ldap.consol.de:389
- **Seit**: 6.8.4

ldap.certificate.searchattr

- **Modul**: cmas-core-server
- **Description**: Name des LDAP-Attributs, mit dem Zertifikate im LDAP-Verzeichnisbaum gesucht werden. Standardwert: mail
- **Typ**: string
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: ja
- **Beispielwert**: mail
- **Seit**: 6.8.4

ldap.certificate.userdn

- **Modul**: cmas-core-server
- **Beschreibung**: DN des LDAP-Zertifikatmanagers. Wenn nicht gesetzt, wird *cmas-core-security, ldap.userdn* verwendet.
- **Typ**: string
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: ja
- **Seit**: 6.8.4

G.2.3.5 Konfiguration des Aktivitätsintervalls

admin.tool.session.check.interval

- **Modul**: cmas-app-admin-tool
- **Beschreibung**: Intervall, in dem inaktive (beendete) Sitzungen im Admin Tool überprüft werden (in Sekunden).
- **Typ**: integer
- **Neustart erforderlich**: ja
- **System**: ja
- **Optional**: nein
- **Beispielwert**: 30
- **Seit**: 6.7.5

server.session.timeout

- **Modul**: cmas-core-server
- **Beschreibung**: Server-Session-Timeout (in Sekunden) für verbundene Clients. Jeder Client kann dieses Timeout mit benutzerdefinierten Werten mittels seiner ID (ADMIN_TOOL, WEB_CLIENT, WORKFLOW_EDITOR, TRACK (vor 6.8 bitte PORTER verwenden), ETL, REST) überschreiben. Diese wird an den Namen der System-Property angehängt, z. B. *server.session.timeout.ADMIN_TOOL*.
  Siehe auch die Attribute der Seitenanpassung *updateTimeServerSessionActivityEnabled* und *updateTimeServerSessionActivity*, beide vom Typ *cmApplicationCustomization*.
- **Typ**: integer
- **Neustart erforderlich**: nein
- **System**: ja
- **Optional**: nein
- **Beispielwert**: 1800
- **Seit**: 6.6.1, 6.7.1

Detaillierte Erklärung für das Admin Tool:

- server.session.timeout.ADMIN_TOOL
  Definiert den Zeitraum, den der Server eine Session im Admin Tool als gültig betrachtet, wenn keine Aktivität im Admin Tool der Session erfolgt. Das Admin Tool kennt diesen Wert nicht, es bemerkt nur eine ungültige Session, wenn es länger keine Aktivität gegeben hat.
- admin.tool.session.check.interval
  Definiert den Zeitraum zwischen zwei Überprüfungen durch das Admin Tool, ob der Server die Session noch als gültig betrachtet.

Wenn zum Beispiel admin.tool.session.check.interval = 60, fragt das Admin Tool den Server einmal pro Minute, ob die Session noch aktiv/gültig ist. Wenn server.session.timeout.ADMIN_TOOL = 600 erhält das Admin Tool die Antwort, dass die Session ungültig ist, nach zehn Minuten der Inaktivität.

G.2.3.6 Administrator-E-Mail-Adressen

ConSol CM kann je nach Untersystem verschiedene Administrator-E-Mail-Adressen verwenden. Eine detaillierte Erklärung über die Administrator-E-Mail-Adressen finden Sie in Administrator-E-Mail-Adressen. Wenn keine speziellen Administrator-E-Mail-Adressen konfiguriert sind, wird die globale E-Mail-Adresse (die Sie bei der Systemeinrichtung angegeben haben), verwendet.

G.3 Administrator-E-Mail-Adressen

In diesem Kapitel werden folgende Themen behandelt:

G.3.1 Einleitung

In ConSol CM können mehrere Administrator-E-Mail-Adressen (oder Benachrichtigungs-E-Mail-Adressen) konfiguriert werden. Hier finden Sie eine Übersicht über alle diese Adressen.

G.3.2 Standardkonfiguration

Wenn Sie ein ConSol CM-System aufsetzen, müssen Sie eine globale Administrator-E-Mail-Adresse eingeben.

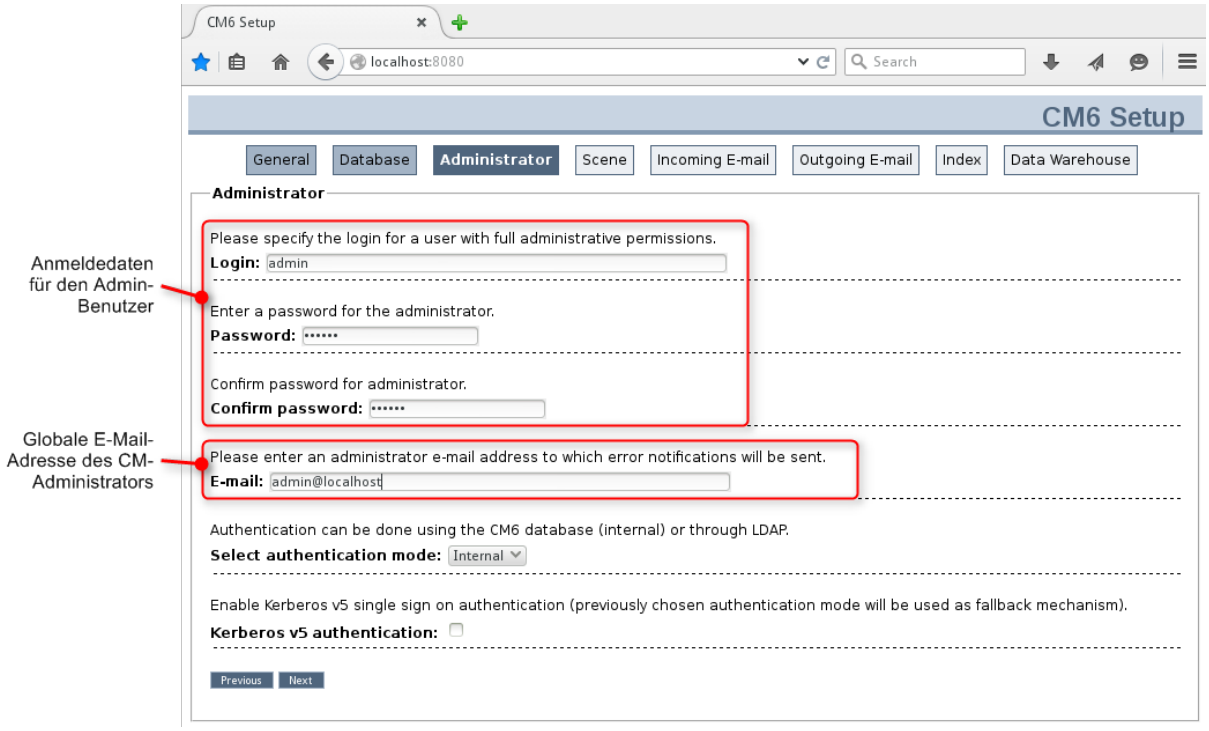


Abbildung 178: *Konfiguration der Administrator-E-Mail-Adresse bei der Systemeinrichtung*

Diese globale E-Mail-Adresse (System-Property *cmas-core-security*, *admin.e-mail*) wird für alle Benachrichtigungen verwendet und automatisch auch für alle spezifischen E-Mail-Adressen der Untersysteme eingetragen. Das bedeutet, dass diese E-Mail-Adresse initial automatisch für alle System-Properties gesetzt wird, die Administrator- oder Benachrichtigungs-E-Mail-Adressen enthalten. Die Properties können über die GUI des Admin Tools geändert werden, um das jeweilige Untersystem zu konfigurieren. Sie können zum Beispiel im Abschnitt *DWH-Konfiguration* des Admin Tools eine spezielle Benachrichtigungs-E-Mail-Adresse für DWH-Operationen konfigurieren.

Durch die Konfiguration von unterschiedlichen Administrator-/Benachrichtigungs-E-Mail-Adressen für unterschiedliche Untersysteme können Sie die Verantwortlichkeiten entsprechend den Zuständigkeiten und Rollen innerhalb Ihres Unternehmens verteilen. Für einige Benachrichtigungen können die From-Adresse, der Text und der Betreff konfiguriert werden.

G.3.3 Spezifische Benachrichtigungs-E-Mail-Adressen für einzelne Untersysteme

G.3.3.1 DWH (Data Warehouse) - Spezifische Benachrichtigungs-E-Mail-Adressen und E-Mail-Konfigurationen

System-Properties

- **Fehler**
  - **cmas-dwh-server, notification.error.to**
    Wenn eine DWH-Operation fehlgeschlagen ist, wird eine E-Mail an diese Adresse gesendet. Wenn die Property nicht gesetzt ist, wird keine E-Mail gesendet.
  - **cmas-dwh-server, notification.error.from**
    Wenn eine DWH-Operation fehlgeschlagen ist, wird eine E-Mail von dieser From-Adresse gesendet.
  - **cmas-dwh-server, notification.error.subject**
    Betreff für E-Mails mit Fehlermeldungen des DWH.
  - **cmas-dwh-server, notification.error.description**
    Text für E-Mails mit Fehlermeldungen des DWH.
- **Erfolgreich**
  - **cmas-dwh-server, notification.finished_successfully.to**
    Es wird eine E-Mail an diese E-Mail-Adresse gesendet, wenn eine DWH-Übertragung erfolgreich abgeschlossen wurde, z. B. wenn eine Übertragung ohne Fehler abgeschlossen wurde. Wenn die Property nicht gesetzt ist, wird keine E-Mail gesendet.
  - **cmas-dwh-server, notification.finished_successfully.from**
    From-Adresse für E-Mails des DWHs, wenn eine Übertragung erfolgreich beendet wurde.
  - **cmas-dwh-server, notification.finished_successfully.subject**
    Betreff für E-Mails des DWHs, wenn eine Übertragung erfolgreich beendet wurde.
  - **cmas-dwh-server, notification.finished_successfully.description**
    Text für E-Mails des DWHs, wenn eine Übertragung erfolgreich beendet wurde.
- **Fehlgeschlagen**
  - **cmas-dwh-server, notification.finished_unsuccessfully.to**
    Es wird eine E-Mail an diese E-Mail-Adresse gesendet, wenn eine DWH-Übertragung abgeschlossen wurde, aber nicht erfolgreich war, z. B. wenn eine Übertragung mit Fehlern abgeschlossen wurde. Wenn die Property nicht gesetzt ist, wird keine E-Mail gesendet.
  - **cmas-dwh-server, notification.finished_unsuccessfully.from**
    From-Adresse für E-Mails des DWHs, wenn eine Übertragung nicht erfolgreich beendet wurde.
  - **cmas-dwh-server, notification.finished_unsuccessfully.subject**
    Betreff für E-Mails des DWHs, wenn eine Übertragung nicht erfolgreich beendet wurde.
  - **cmas-dwh-server, notification.finished_unsuccessfully.description**

Text für E-Mails des DWHs, wenn eine Übertragung nicht erfolgreich beendet wurde.

- **cmas-dwh-server, cmas-dwh-server, notification.error.description**
  Text für E-Mails mit Fehlermeldungen des DWH.

Eine Übersicht mit allen System-Properties, die für DWH-Benachrichtigungen gesetzt werden können, finden Sie im Abschnitt CMRF- und DWH-Konfiguration des Kapitels Liste der System-Properties nach Bereich. Alle Properties, die in diesem Kontext relevant sind, beginnen mit *notification.*.

Grafische Konfiguration

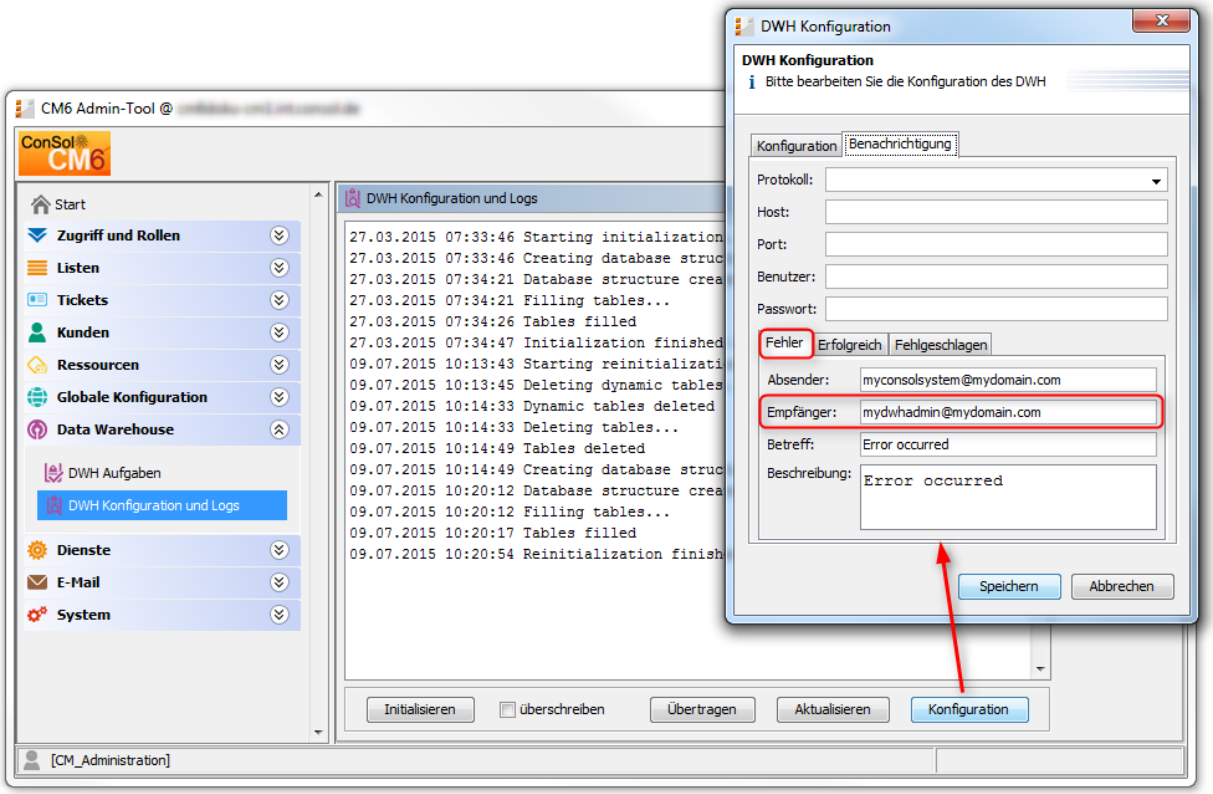


Abbildung 179: *ConSol CM Admin Tool: Benachrichtigungs-E-Mail-Adresse für DWH-Fehler*

Die E-Mail-Adresse wird überprüft, wenn Sie auf *Speichern* klicken. Wenn die E-Mail-Adresse nicht gültig ist, wird eine Meldung angezeigt. Es werden keine E-Mails an diese E-Mail-Adresse gesendet.

G.3.3.2 E-Mail - Spezifische Benachrichtigungs-E-Mail-Adressen

System-Properties (NIMH-Modus)

- **cmas-nimh-extension, mail.error.to.address**
  Im Fall, dass eine E-Mail nicht verarbeitet werden konnte, wird eine Fehler-E-Mail an die E-Mail-Adresse gesendet. Ab CM-Version 6.10.5.1 wird auch bei E-Mail-Timeouts eine E-Mail an diese Adresse gesendet.
  Wenn die Property nicht gesetzt ist, wird keine E-Mail gesendet und es wird eine Exception in die Log-Datei geschrieben.

- **cmas-nimh-extension, mail.attachments.validation.info.sender**
  Setzt den From-Header bei Attachments vom Typ *error notification e-mail*.

Beachten Sie, dass das Versenden von der System-Property *cmas-nimh-extension*, *mail.on.error* gesteuert wird. Nur wenn diese Property auf *true* gesetzt ist, wird im Fall, dass eine E-Mail nicht verarbeitet werden konnte, ein Fehler-E-Mail an die oben konfigurierte E-Mail-Adresse gesendet.

System-Properties (Mule/ESB-Modus)

- **cmas-esb-mail, mail.process.error**
  To-Adresse für E-Mails mit Fehlermeldungen des Mule/ESB-Mail-Service. Im Fall, dass eine E-Mail nicht verarbeitet werden konnte, es ein Problem mit dem Status gibt oder andere Probleme mit dem Mule/ESB-Mail-Service aufgetreten sind, wird eine Fehler-E-Mail an die E-Mail-Adresse gesendet. Wenn die Property nicht gesetzt ist, schlägt das Versenden der E-Mail fehl und dies wird protokolliert.
- **cmas-esb-mail, mail.mule.service**
  From-Adresse für E-Mails, die vom Mule/ESB-Mail-Service gesendet werden.

Grafische Konfiguration

Der Wert, der hier auf der grafischen Benutzeroberfläche eingegeben wird (*Administrator E-Mail* im Abschnitt *Konfiguration*), wird sowohl für NIMH als auch für ESB/Mule verwendet.

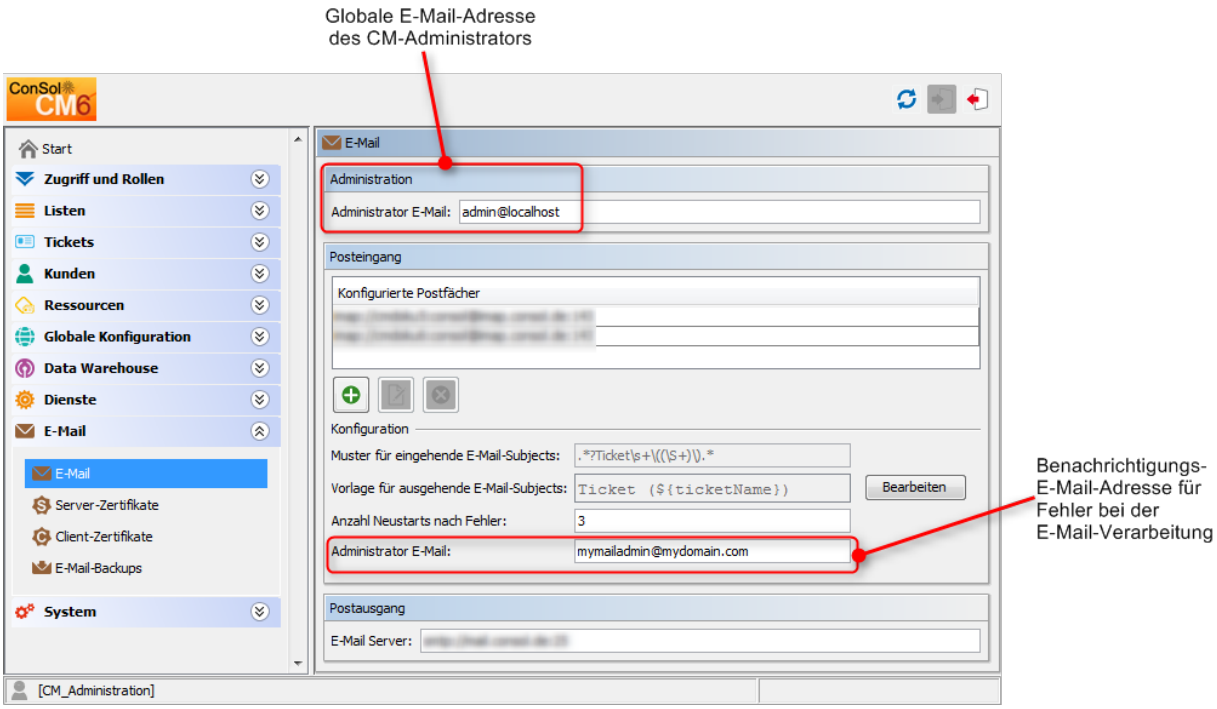


Abbildung 180: *ConSol CM Admin Tool: Konfiguration der globalen Administrator-E-Mail-Adresse und Benachrichtigungs-E-Mail-Adresse für E-Mail-Verarbeitungsprobleme*

Verwechseln Sie die beiden E-Mail-Adressen, die im Tab des Navigationselements *E-Mail* konfiguriert werden können, nicht!

- **Administrator E-Mail** unter **Administration**
  ist die globale E-Mail-Adresse des Administrators (bei der Einrichtung angegeben).
- **Administrator E-Mail** unter **Konfiguration**
  ist die E-Mail-Adresse des E-Mail-Administrators oder einer anderen Person, die die Fehlermeldungen erhalten soll, die auftreten, wenn die Verarbeitung der (eingehenden oder ausgehenden) E-Mails nicht richtig funktioniert.

G.3.3.3 Workflow-Engine - Spezifische Benachrichtigungs-E-Mail-Adressen

- **cmas-workflow-engine, jobExecutor.adminMail**
  E-Mail-Adresse, an die Benachrichtigungs-E-Mails, die Probleme der Jobausführung betreffen, geschickt werden (wenn die Anzahl der Neuversuche überschritten wurde).
- **cmas-workflow-engine, jobExecutor.mailFrom**
  E-Mail-Adresse, die als From-Header für Admin-Benachrichtigungen eingesetzt wird.

G.4 Liste der Code-Beispiele

In diesem Abschnitt finden Sie eine Liste der Code-Beispiele aus diesem Handbuch.

G.5 Marken

- Apache OpenOffice™ – Apache und die Apache-Federlogos sind Marken von The Apache Software Foundation. OpenOffice.org und das Mövenlogo sind eingetragene Marken von The Apache Software Foundation. Siehe Website von Apache OpenOffice zu Markenrichtlinien.
- Google Maps™ – Google Maps ist eine Marke von Google Inc. Siehe Google-Webseite über Marken.
- Microsoft® – Microsoft und Windows sind entweder eingetragene Marken oder Marken der Microsoft Corporation in den USA und/oder anderen Ländern. Siehe Website von Microsoft zu Markenrichtlinien.
- Microsoft® Active Directory® – Microsoft und Microsoft Active Directory sind entweder eingetragene Marken oder Marken der Microsoft Corporation in den USA und/oder anderen Ländern. Siehe Website von Microsoft zu Markenrichtlinien.
- Microsoft® Exchange Server – Microsoft und Microsoft Exchange Server sind entweder eingetragene Marken oder Marken der Microsoft Corporation in den USA und/oder anderen Ländern. Siehe Website von Microsoft zu Markenrichtlinien.
- Microsoft® Office – Microsoft und Microsoft Office sind entweder eingetragene Marken oder Marken der Microsoft Corporation in den USA und/oder anderen Ländern. Siehe Website von Microsoft zu Markenrichtlinien.
- Windows® Betriebssystem – Microsoft und Windows sind entweder eingetragene Marken oder Marken der Microsoft Corporation in den USA und/oder anderen Ländern. Siehe Website von Microsoft zu Markenrichtlinien.
- Microsoft® SQL Server® – Microsoft und Microsoft SQL Server sind entweder eingetragene Marken oder Marken der Microsoft Corporation in den USA und/oder anderen Ländern. Siehe Website von Microsoft zu Markenrichtlinien.
- Microsoft® Word® – Microsoft und Microsoft Word sind entweder eingetragene Marken oder Marken der Microsoft Corporation in den USA und/oder anderen Ländern. Siehe Website von Microsoft zu Markenrichtlinien.
- MuleSoft™ und Mule ESB™ sind Marken von MuleSoft, Inc. Siehe Website von Mule Soft.
- Oracle® – Oracle ist eine eingetragene Marke von Oracle Corporation und/oder ihren verbundenen Unternehmen. Siehe Website von Oracle zu Markenrichtlinien.
- Oracle® WebLogic – Oracle ist eine eingetragene Marke von Oracle Corporation und/oder ihren verbundenen Unternehmen. Siehe Website von Oracle zu Markenrichtlinien.
- Pentaho® – Pentaho und das Pentaho-Logo sind eingetragene Marken von Pentaho Inc. Siehe Website von Pentaho zu Markenrichtlinien.

G.6 Glossar

A

ACF

ACF ist die Abkürzung von Activity Control Form (auf Deutsch Aktivitäts-Formular). ACFs können in Workflow-Aktivitäten verwendet werden, um den Bearbeiter zu zwingen, bestimmte Datenfelder auszufüllen, bevor er fortfahren kann.

ACIM

Activity Item - Eintrag im Ticketbereich Protokoll eines Tickets (z.B. Kommentar, E-Mail, Attachment, Zeitbuchungseintrag).

AD

Microsoft Active Directory - ein LDAP-basierter Verzeichnisservice für Microsoft Windows-Domänennetzwerke.

Admin Tool

ConSol CM-Kompenente, grafische Applikation, um ein ConSol CM-System zu konfigurieren und zu verwalten. Verwendet Java Web Start.

B

Bearbeiter

Bearbeiter sind die Benutzer, die im Web Client an Tickets arbeiten.

Benutzerdefinierte Feldgruppe

Eine Gruppe von Benutzerdefinierten Feldern, in denen Ticketdaten gespeichert werden.

Benutzerdefiniertes Feld

Ein Feld, in dem Ticketdaten gespeichert werden.

Berechtigung

Mit Berechtigungen wird festgelegt, welche Tickets der Bearbeiter im Web Client sehen kann und welche Aktionen er durchführen darf. Berechtigungen werden immer über Rollen erteilt, d.h. sie werden nicht einem einzelnen Benutzer zugewiesen, sondern einer Gruppe von Benutzern, die die gleiche Rolle haben. Normalerweise gehören diese Benutzer zum gleichen Team und/oder haben ähnliche Funktionen im Unternehmen.

BI

Business Intelligence - Methoden, Technologien und Architekturen, um Daten in für Geschäftszwecke nützliche Informationen umzuwandeln.

C

CFEL

Custom Field Expression Language - Java-Klassen und -Methoden des ConSol CM APIs, um auf Daten von Benutzerdefinierten Feldern, Date-

nobjektgruppenfeldern und Ressourcenfeldern zuzugreifen.

CM.Doc

Ein ConSol CM-Standard-Modul, das es Bearbeitern ermöglicht, über den Web Client mit Microsoft Word- oder OpenOffice-Dokumenten zu arbeiten, die mit ConSol CM-Ticket- oder Kundendaten vorausgefüllt sind.

CM.Resource Pool

CM.Resource Pool ist ein optionales Add-on, das es ermöglicht, unterschiedliche Arten von Objekten als Ressourcen in ConSol CM zu speichern.

CM.Track

CM.Track ist das Portal von ConSol CM. Kunden erhalten über CM.Track Zugriff auf ihre Tickets.

CMDB

ConSol CM-Datenbank - die Arbeitsdatenbank des CM-Systems.

CMRF

ConSol CM Reporting Framework - eine Java EE-Applikation, die Daten zwischen der ConSol CM-Datenbank und dem DWH synchronisiert.

CTI

Computer Telephony Integration - übergeordnete Bezeichnung für Technologien, die eine Verknüpfung von Telefonanlagen mit Computersystemen ermöglichen.

D

Datenobjekt

Ein Kunde (Kontakt oder Firma). Ehemals Unit.

Datenobjektgruppe

Eine Gruppe von Feldern, in denen Daten für Kunden (Kontakte oder Firmen) gespeichert werden. Ähnlich den Benutzerdefinierten Feldgruppen für Ticketdaten.

Datenobjektgruppenfeld

Ein Feld, in dem Daten für Kunden (Kontakten oder Firmen) gespeichert werden. Ähnlich den Benutzerdefinierten Feldern für Ticketdaten.

DWH

Data Warehouse - ConSol CM-Datenbank, die für Reporting und Datenanalyse verwendet wird.

E

ERP-System

Enterprise Resource Planning - häufig verwendet für diese Art von Enterprise Management Software.

ESB

Enterprise Service Bus - eine Software-Architektur, die für die Kommunikation zwischen gemeinsam interagierenden Software-Applikationen in einer service-orientierten Architektur (SOA) verwendet wird.

ETL

Extract Transform Load - extrahiert Daten aus einer Quelle (dies kann eine Datenbank oder eine andere Quelle sein), wandelt diese um und lädt sie in eine andere Datenbank, z.B. ein Data Warehouse.

F

Firma

Die Firma stellt die obere Hierarchiestufe in einem zweistufigen Kundenmodell dar. Eine Firma kann mehrere Kontakte haben.

FlexCDM

Das Flexible Kundendatenmodell - das Kundendatenmodell, welches in ConSol CM-Version 6.9 eingeführt wurde. Für jede Kundengruppe kann ein eigenes Kundendatenmodell definiert werden.

G

GUI

Graphical User Interface

H

Hauptkunde

Der Hauptkunde ist der Kunde, der der Grund für die Erstellung des Tickets ist. In einem Ticket muss ein Hauptkunde angegeben sein.

I

IMAP

Internet Message Access Protocol - Internet-Standardprotokoll für den Zugriff auf E-Mails auf einem Remote-E-Mail-Server. Kann als einfaches (plain) IMAP oder sicheres (secure) IMAP (IMAPs) verwendet werden. Im letzteren Fall wird ein gültiges Zertifikat benötigt.

J

Java EE

Java Enterprise Edition

JMS

Java Message Service - Java EE Komponente, um Nachrichten zwischen JMS-Clients zu versenden.

JRE

Java Runtime Environment. Java-Laufzeitumgebung, enthält die virtuelle Maschine, die für die Ausführung von Java-Anwendungen nötig ist.

K

Kontakt

Der Kontakt stellt die untere Hierarchiestufe in einem zweistufigen Kundenmodell dar. Ein Kontakt kann nur zu einer Firma gehören.

KPI

Key Performance Indicator - Parameter für die Performance-Messung für Firmen, Projekte usw.

Kunde

Der Kunde stellt die externe Seite eines Tickets dar. Er ist die Person oder das Objekt, das den Grund für die Erstellung eines Tickets bildet. Ein Kunde kann entweder eine Firma oder ein Kontakt sein.

Kundenaktion

Teil des Action Frameworks - Aktion, die für ein Kundenobjekt (d.h. eine Firma oder einen Kontakt) ausgeführt wird.

Kundendatenmodell

Das Kundendatenmodell ist die Definition der Kunden. Es bestimmt die verfügbaren Datenfelder und möglichen Relationen.

Kundengruppe

Die Kundengruppe bestimmt, welches Kundendatenmodell für ihre Kunden verwendet wird und welche Aktionen verfügbar sind.

L

LDAP

LDAP ist die Abkürzung für Lightweight Directory Access Protocol. Das ist ein Protokoll, mit dem Anmeldeinformationen für mehrere Applikationen verwaltet werden.

LDAPS

LDAP über SSL

M

Mule

Ein Open Source Java-basierter Enterprise Service Bus (ESB).

N

NIMH

New Incoming Mail Handler - Modul für das Abrufen von eingehenden E-Mails, neu seit Version 6.9.4.

P

PCDS

Page Customization Definition Section - Definitionsbereich der Seitenanpassung.

Pentaho

PentahoTM ist eine Business Intelligence-Produktsuite, die als Open-Source- und als Enterprise-Version verfügbar ist.

POP

Post Office Protocol - Standard-Internetprotokoll für den Fernzugriff auf einen E-Mail-Server mittels TCP/IP. Kann als einfaches POP oder als verschlüsseltes POP (POPS) benutzt werden. Im letzteren Fall werden die geeigneten Zertifikate benötigt.

Portal

CM.Track - ermöglicht Kunden Zugriff auf ConSol CM.

Postfach

Zielort, an den E-Mails zugestellt werden. Postfächer werden auf einem E-Mail-Server verwaltet. ConSol CM kann E-Mails von einem oder mehreren getrennten Postfächern abrufen.

Process Designer

ConSol CM-Komponente für das Design, die Erstellung und das Installieren von Workflows.

Protokoll

Das Protokoll enthält alle Änderungen, die an dem Ticket, Kunden oder der Ressource ausgeführt wurden.

Q

Queue

Die Queue enthält thematisch ähnliche Tickets, die gleich behandelt werden sollen und dem gleichen Geschäftsprozess (Workflow) folgen. Berechtigungen und andere Parameter werden immer auf der Basis von Queues definiert.

R

RDBMS

Relational Database Management System (relationales Datenbankmanagementsystem) - z.B. Oracle ® , MS SQL Server ® , MySQL.

Relation

Relationen sind Verknüpfungen zwischen unterschiedlichen Datenobjekten in ConSol CM. Es gibt Relationen zwischen Objekten des gleichen Typs, z.B. zwischen Tickets, Kunden und Ressourcen, und Relationen zwischen Objekten unterschiedlicher Typen, z.B. zwischen einem Ticket und einer Ressource oder einem Kunden und einer Ressource.

Ressource

Ressourcen sind Objekte, die in CM.Resource Pool verwaltet werden.

Ressourcenaktion

Teil des Action Frameworks - Aktion, die für ein Ressourcenobjekt ausgeführt wird.

Ressourcenfeld

Feld, in dem Ressourcendaten gespeichert werden.

Ressourcenfeldgruppe

Eine Gruppe von Feldern, in denen Daten für Ressourcen gespeichert werden. Ähnlich den Benutzerdefinierten Feldgruppen für Ticketdaten.

Ressourcentyp

Der Ressourcentyp ist die Definition der Ressourcen. Es bestimmt die verfügbaren Datenfelder und möglichen Relationen und Aktionen.

REST

Representational State Transfer - Methode, um Daten mittels eines Netzwerks zu übermitteln, basierend auf HTTP.

Rolle

Rollen werden den Bearbeitern zugewiesen. Sie bestimmen die Zugangsberechtigungen und Sichten der Bearbeiter.

S

Sicht

Sichten beschränken die in der Ticketliste im ConSol CM Web Client angezeigten Tickets auf die Tickets, die bestimmte Kriterien (Bereiche eines oder mehrerer Workflows) erfüllen. Sichten werden Rollen zugewiesen.

Skript

Programme, die für eine spezielle Laufzeitumgebung geschrieben werden, die die Ausführung von Aufgaben interpretieren und automatisieren können. In ConSol CM werden Skripte im Admin Tool gespeichert und als Skripte für Aktivitäten in Workflows gespeichert.

SMTP

Simple Message Transfer Protocol - Standard-Protokoll für den Versand von E-Mails.

Suche-Aktion

Teil des Action Frameworks - Aktion, die für das Ergebnis einer Suche ausgeführt wird.

T

TAPI

Telephony Application Programming Interface - Microsoft Windows API, welche Computer-Telefonie-Integration bereitstellt und es PCs unter Microsoft Windows ermöglicht, Telefondienste zu verwenden.

TEF

Task Execution Framework - ein ConSol CM-Modul, das Aufgaben (Tasks) asynchron ausführen kann. Neu seit Version 6.9.4.

Ticket

Das Ticket ist die Kundenanfrage, an der der Bearbeiter arbeitet. Es ist das Objekt, das den vom Workflow definierten Geschäftsprozess durchläuft.

U

Unit

Java-Klasse, die einen Kunden darstellt. Ein Kontakt ist ein Objekt der Klasse Unit, und eine Firma ist ein Objekt der Klasse Unit.

V

Vorlage

Vorlagen enthalten vordefinierten und vorformatierten Text. Sie können für Kommentare, E-Mails und Dokumente verwendet werden.

W

Web Client

Der Web Client ist der Hauptzugang zum System für die Bearbeiter.

Workflow

Der Workflow ist die Umsetzung des in ConSol CM verwalteten Geschäftsprozesses. Er enthält eine Reihe von Schritten, die von den Bearbeitern durchgeführt werden.

Z

Zeitbuchung

Zeitbuchungen ermöglichen es den Bearbeitern, die Zeit festzuhalten, die sie an einem Ticket oder Projekt gearbeitet haben.

Zusatzkunde

Zusatzkunden sind Kunden (Firmen oder Kontakte), die am Ticket interessiert sind. Sie sind optional und haben normalerweise eine Rolle, die anzeigt, wieso sie hinzugefügt wurden.

zusätzlicher Bearbeiter

Zusätzliche Bearbeiter sind Bearbeiter, die einen bestimmten, im Geschäftsprozess definierten Zweck erfüllen. Normalerweise müssen sie im Prozess bestimmte Aufgaben ausführen.